Visual Studio 2005 Visual J#

Visual J#

Microsoft Visual J# 2005 は、Java 言語構文を使用する開発者が .NET Framework 上で動作するアプリケーションやサービスを作成するためのツールです。Visual J# では、Java 言語構文は Visual Studio® 統合開発環境 (IDE: Integrated Development Environment) に統合されています。また、Visual J# は、Microsoft 拡張機能を含む Visual J++ 6.0 のほとんどの機能をサポートしています。Visual J# は、Java 仮想マシン上で実行することを目的としたアプリケーションの開発ツールではありません。Visual J# を使って作成したアプリケーションとサービスは、.NET Framework だけで動作します。Visual J# は、Microsoft が独自に開発したものです。Visual J# は、Sun Microsystems, Inc. によって承認された言語ではありません。

Visual J# は Visual Studio IDE に統合されているため、Visual J# のプログラマは Visual Studio デザイナを使用して XML Web サービス、Web フォーム ページ、および Windows フォーム アプリケーションを作成できます。

このセクションの内容

Visual J# の概要

Visual J# プロジェクトの作成を開始する上で役立つトピックの一覧を示します。

Visual J++ 6.0 からのアップグレード

Visual J++ 6.0 プロジェクトを Visual J# にアップグレードする場合の推奨事項と手順について説明します。

Visual J# の移行

Java アプリケーションおよび J++ アプリケーションを Visual J# に変換する際に使用できる便利なツールとウィザードについて説明します。

Visual J# アプリケーションの配置

Visual J# での配置に固有の問題について説明します。配置とは、完成したアプリケーションまたはコンポーネントを配布し、他のコンピュータにインストールするプロセスです。

Visual J# のタスク

Visual J# を使用した共通のプログラミング タスクの実行方法を説明した "方法" トピックの一覧を示します。

Visual J# リファレンス

Visual J# コンパイラ、バイナリ変換ツール (JbImp.exe)、言語とライブラリのサポート、Visual J++ 6.0 プロジェクトのアップグレード、.NET Framework クラス ライブラリ用の Visual J# 構文、および開発環境の機能に関するリファレンス トピックです。

Visual J# のサンプル

Visual J# を使って .NET Framework 用のアプリケーションを記述する方法を示すサンプル ソース コードです。

Visual J# のチュートリアル

分散 Web アプリケーションを作成する方法、コンポーネントとコントロールを編集する方法、Windows フォームと Web フォームを操作する方法、およびデータ関連のタスクを実行する方法を段階的に説明しているトピックの一覧です。

関連するセクション

Visual Studio

Visual Studio ドキュメントの主な開始点へのリンクを示します。

Windows フォーム

Windows ベースのアプリケーションを作成するためのテクノロジとツールに関するトピックへのリンクがあります。

Web フォーム ページ

ASP.NET アプリケーションでブラウザ ベースのユーザー インターフェイスを作成するためのテクノロジやツールに関するトピックへのリンクの一覧です。

マネージ コードを使用した XML Web サービス

XML Web サービスを作成、配布、および利用するためのテクノロジについて説明します。

Visual Studio でのデバッグ

Visual Studio 環境でのデバッグについて説明します。

Visual J# の概要

このセクションでは、Visual J# 開発システムの機能、特徴、およびアーキテクチャの概要について説明します。

このセクションの内容

Visual J# の概要

Visual J# の利点、機能、および主な特徴の概要について説明します。

Visual J# 2005 の新機能

Visual J# の新しい機能について説明します。

Visual J# 言語の新機能

J# 言語の新しい機能について説明します。

Visual J# のアーキテクチャ

Visual J# のコンパイラ、拡張機能、およびバイナリ コンバータが、J# クラス ライブラリ、.NET Framework、および共通言語ランタイムに対してどのように設計されているかについて説明します。

開発者情報

Visual J# とその使い方、および Java 言語との関係について簡単に説明します。

Visual J# の設定

Visual Studio IDE をカスタマイズするための Visual J# 設定について説明します。

関連するセクション

Visual J#

Visual J# ドキュメントのさまざまなトピックへのリンクの一覧です。

Visual J# の概要

Visual J# は、Java 言語のプログラマが .NET Framework で動作するアプリケーションやサービスを作成するために使用できるツールです。

Visual J# では共通言語ランタイム (CLR: Common Language Runtime) を使っており、XML Web サービス、Web アプリケーションなど、.NET Framework を最大限に利用する .NET Framework アプリケーションを開発できます。Visual J# では、次のような便利な機能がサポートされています。

- 言語間での統合
- 強化されたセキュリティ
- バージョン管理と配置のサポート
- デバッグ サービスとプロファイル サービス

メモ：
Visual Studio® を使うと、コンピュータに Visual J# をインストールしていなくても Java 言語アプリケーションをデバッグできます。

Visual J# の既定のソース ファイル拡張子は .jsl です。Visual J# には次のものが付属します。

- Java 言語のソースを Microsoft® Intermediate Language (MSIL) にコンパイルする Visual J# コンパイラ。詳細については、「Visual J# コンパイラ」を参照してください。
- Java 言語のバイトコードを MSIL に変換するバイナリ コンバータ。詳細については、「Visual J# バイナリ変換ツール (Java 言語のバイトコードから MSIL への変換用)」を参照してください。
- 独自に開発されたクラス ライブラリ。これらは、JDK レベル 1.1.4 のほとんどのクラス ライブラリに相当する機能と、College Board の Advanced Placement カリキュラムの Computer Science で指定されている JDK 1.2 java.util パッケージのクラスの多くに相当する機能をサポートするように設計されています。詳細については、「クラス ライブラリのサポート」および「Visual J# Class Library」を参照してください。
- WFC (Windows® Foundation Classes) および多くの com.ms.* パッケージとの互換性。詳細については、「サポートされているクラス ライブラリ」を参照してください。
- Visual J# コンパイラは、Visual Studio 2005 で 32 ビット アプリケーションだけを作成できます。64 ビットの Windows オペレーティング システムで実行する場合、Microsoft Visual J# 2005 でコンパイルしたアプリケーションは WOW64 (Windows on Windows64) で実行します。64 ビット アプリケーションの詳細については、「64 ビット アプリケーション」を参照してください。

ただし、Visual J# には次の制限があります。

- Java 言語のソース コードは、Java 言語バイトコード、つまり .class ファイルにはコンパイルされません。
- Java 仮想マシン上で実行されるアプリケーションを作成する機能はサポートされていません。
- JNI (Java Native Interface)、RNI (Raw Native Interface)、および RMI (Remote Method Invocation) はサポートされていません。

Microsoft Visual J# 2005 は、Java 仮想マシン上で実行されるアプリケーションの開発には使用されません。Visual J# を使って作成したアプリケーションとサービスは、.NET Framework だけで動作します。Visual J# は、Microsoft が独自に開発したものです。Visual J# は、Sun Microsystems, Inc. によって承認された言語ではありません。

参照

概念

Visual J# のアーキテクチャ

その他の技術情報

Visual J# の概要

Visual J# 2005 の新機能

Visual J# 2005 には、次の新機能が導入されています。

新機能

- 共通言語仕様への準拠の検証
- 複数のバージョンの共通言語ランタイムで並行に作業を進めるためのサポートが強化されました。
- 例外処理モデルが変更され、**java.lang.Throwable** をキャッチする catch 句で System.Exception から継承した .NET 例外をキャッチできるようになりました。詳細については、「Visual J# の例外の階層構造」を参照してください。
- Visual J# ライブラリに **AllowPartiallyTrustedCallers** 属性が追加されました。これによって、アプレットなど、部分的に信頼されているコードを実行できます。詳細については、「Visual J# で記述されたアプリケーションでのセキュリティのセマンティクス」を参照してください。
- Java bean スタイルのプロパティがサポートされました。詳細については、「bean スタイルのプロパティ」を参照してください。
- CodeDOM を使用したキャストがサポートされました。詳細については、「チュートリアル : Java 言語でのソース コードの動的な生成とコンパイル」を参照してください。
- 一部のクラスが変更され、JDK 1.2 レベル相当の機能がサポートされました。
- 領域を指定しない場合、**java.util.Locale** クラスは既定の領域へのフォールバックを許可しません。この変更により、このフォールバック機構に依存するアプリケーションが動作しない可能性があります。コードがフォールバック機構を使用している場合、*country/region* パラメータは空の文字列で示されます。ただし、新しく追加された **com.ms.vjsharp.text.FormatDefaults.setDefaultRegion** 関数を呼び出すことによってこのコードを動作させることもできます。詳細については、「com.ms.vjsharp.text.FormatDefaults クラスのメソッド」を参照してください。
- 基本データ型の **java.lang** ラッパー クラスは、System.IConvertible をサポートするようになりました。これにより、**System.IConvertible** サポートが必要な場合に、.NET Framework で基本データ型を使用できます。詳細については、「.NET Framework 型と Java ラッパー型の変換」を参照してください。
- 内部例外がサポートされました。詳細については、**java.lang.Throwable** の内部例外のサポートに関するトピックを参照してください。
- Java 言語ソース コードで Javadoc コメントと呼ばれている形式のドキュメント コメントがサポートされました。詳細については、「方法 : Javadoc コメントから XML ドキュメントを生成する」を参照してください。
- シリアル化できる Java 言語の型で、自動的に .NET シリアル化がサポートされるようになりました。詳細については、「.NET シリアル化のサポート」を参照してください。
- Supplemental UI Library (VJSSupUILib.dll) を使用したアプレット開発がサポートされました。
- 新機能を使用した新しい Visual J# アプリケーションのサンプルには、次のものがあります。
    - bean スタイルのインデックス付きプロパティのサンプル (プロパティ値の設定)
    - bean スタイルの簡単なプロパティのサンプル (プロパティ値の設定)
    - 参照渡しのサンプル (型の参照渡し)
    - カスタム属性のサンプル (カスタム属性のクラスへの追加)
    - 列挙型のサンプル (列挙型の宣言)
    - ジェネリックのサンプル (ジェネリックの使用)
    - 値型のサンプル (ポイント型を値型として宣言)

詳細については、「Visual J# 言語の新機能」を参照してください。

参照

概念

Visual Studio 2005 の新機能

Visual J# 言語の新機能

Visual J# 2005 では、言語とコンパイラに次の変更が加えられました。

Visual J# 言語

- カスタム属性の作成
- ユーザー定義の列挙型 (J#)
- ユーザー定義の値型
- 参照渡しの引数を受け取るメソッドの定義
- J# 言語でのジェネリック型とジェネリック メソッドを使用する機能。詳細については、「J# のジェネリック」を参照してください。
- Java 言語の **strictfp** キーワードのサポート。
- **#line hidden** ディレクティブのサポート。**#line hidden** ディレクティブは、後続の行をデバッガで非表示にします。

Visual J# コンパイラ

- 新しいコンパイラは高度なアセンブリ署名に対応するようになりました。詳細については、「/delaysign (暗号化キーの遅延署名)」、「/keycontainer (暗号化キー コンテナの指定)」、および「/keyfile (暗号化キーのファイル名の指定)」を参照してください。
- Visual J# コンパイラ オプションに対応する Visual J++ コンパイラ オプション
- コンパイラは 共通言語仕様への準拠の検証 をサポートするようになりました。
- コンパイラは、bean スタイルのプロパティ、カスタム属性、ジェネリック、アセンブリ、および CLS 準拠に関する新しい診断エラー メッセージを含みます。

参照

概念

Visual J# 2005 の新機能

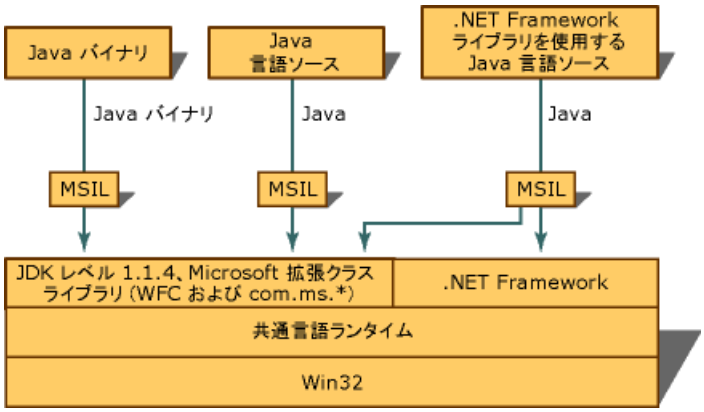
その他の技術情報

Visual J# の概要

Visual J# のアーキテクチャ

Visual J# には次のものが付属します。

- Visual J# コンパイラ (vjc.exe)
- Visual J# バイナリ変換ツール (JbImp.exe)
- 独自に開発されたクラス ライブラリのセット。これは、JDK レベル 1.1.4 のほとんどのクラス ライブラリの機能と、College Board の Advanced Placement カリキュラムの Computer Science で指定されている JDK 1.2 java.util パッケージのクラスの多くをサポートするように設計されています。
- Visual J++ 6.0® の Microsoft® 拡張機能のサポート。これには、WFC (Windows® Foundation Classes) や、多くの com.ms.* パッケージが含まれます。
- Visual J# のクラス ライブラリは、次の図に示すように、.NET Framework および共通言語ランタイムの最上位部分に階層化されています。



Visual J++ 6.0 で開発されたほとんどの Java 言語アプリケーションのソースは、Visual J# コンパイラを使って .NET マネージ実行可能ファイルにコンパイルできます。バイトコード形式でしか使用できない一部のアプリケーションは、Visual J# バイナリ変換ツールによって .NET マネージ実行可能ファイルに静的に変換できます。新しい Visual J# アプリケーションを開発して、J# クラス ライブラリ (JDK レベル 1.1.4 仕様のサブセットの Microsoft による実装)、Visual J++ 6.0 の Microsoft 拡張機能 (WFC や com.ms.* など)、および .NET Framework クラス ライブラリを利用することもできます。

参照

概念

Visual J# の概要

その他の技術情報

Visual J# の概要

Visual J# と Java の関係

Microsoft Visual J# 2005 は、Java 言語構文に慣れている開発者が .NET Framework 上でアプリケーションやサービスを構築するために使用できる開発ツールです。Visual J# では、Java 言語構文が Visual Studio シェルに統合されています。また、Visual J# は、Microsoft 拡張機能を含む Visual J++ 6.0 の機能をサポートしています。Visual J# アプリケーションを開発する場合には、次の点を考慮します。

- Visual J# アプリケーションは、Java 仮想マシン上で実行できません。Visual J# を使って作成したアプリケーションとサービスは、.NET Framework だけで動作します。
- Microsoft Visual J# 再頒布可能パッケージにより、適切なランタイム ライブラリを提供することで Microsoft Visual J# 2005 アプリケーションを提供できます。この再頒布可能パッケージは、Visual J# で作成されたアプリケーションやサービスだけを実行します。他の Java 言語開発ツールで作成された Java 言語アプリケーションは、Microsoft Visual J# 再頒布可能パッケージでは実行できません。

Visual J# および Microsoft Visual J# 再頒布可能パッケージは、Microsoft が独自に開発したもので、Sun Microsystems, Inc. によって承認されたものではありません。

参照

関連項目

Visual J# 再頒布可能パッケージの配布

その他の技術情報

Visual J# の概要

Visual J# の設定

Visual J# の設定は、操作性を最適化し、Visual J# を使った開発の生産性を最大化するように設計されています。これらの設定によって、Visual Studio 統合開発環境 (IDE: Integrated Development Environment) の設定をカスタマイズして保存できます。

また、IDE の設定を他のコンピュータに移植したり、再読み込みしたりできます。詳細については、「方法 : コンピュータ間で設定を共有する」を参照してください。

ウィンドウとビュー

機能	既定での表示	メモ
クラス ビュー	○	• フィルタ処理を有効にします。 • クラス ビュー フォルダを有効にします。 • クラスの [実装されたインターフェイス] ノードを非表示にします。
コマンド ウィンドウ	×	ユーザー オプションとして表示されます。
[ダイナミック ヘルプ] ウィンドウ	×	現在、F1 キーを押しても使用できません。
オブジェクト ブラウザ	○ (物理ビューのみ)	• 継承したメンバを非表示にします。 • 既定で、すべてのクラス メンバを表示します。
[出力] ウィンドウ	○ (ビルドの開始時のみ)	ショートカット キーで GotoNextError コマンドを呼び出します。
ソリューション エクスプローラ	○	既定で、コード エディタとテキスト エディタの [選択したファイルと開いたファイルとを同期する] をオフにします。
[スタート ページ]	○	IDE を初めて起動するときだけに表示されます。
タスク一覧 (Visual Studio)	×	ビルド コマンドによって開きません。
ツールボックス	○	コンポーネントをアルファベット順に表示します。

また、次の要素には、Visual J# プロファイルを選択したときに有効になる特定の動作があります。

ダイアログ ボックス

機能	動作
[クイック検索] ([検索と置換] ウィンドウ)	特定の文字列を 1 つまたは複数置き換えます。
[新しいプロジェクト] ダイアログ ボックス	コンピュータに他のプラットフォームがインストールされている場合、操作対象のプラットフォームを指定するコンボ ボックスを有効にします。
[オプション] ダイアログ ボックス (Visual Studio)	[その他のオプション] ボタンをクリックすると、ソース管理、デバイス ツール、データベース ツール、HTML デザイナ、および XML デザイナのオプションが表示されます。

キーボード

機能	動作
ショートカット キー	• 標準の Visual Studio エディタ • BRIEF エミュレーション • EMACS エミュレーション

その他の IDE 要素

機能	動作
Windows フォーム デザイナの設定	既定のビューはコード エディタです。コンポーネント デザイナ ビューではありません。
メイン メニュー、ショートカット メニュー、ツール バー、コマンド	対応するショートカット キーストロークがメイン メニュー項目の横に表示されます。
シェルの外観	Visual Studio に共通する外観と操作性を持ちます。
ウィンドウ管理とレイアウト	ウィンドウの既定のレイアウトは次のとおりです。 ● ソリューション エクスプローラ、クラス ビュー、およびリソース ビューのツール ウィンドウは右側に、最大の長さで表示されます。 ● 出力ウィンドウとプロパティ グリッドは、既定で表示されます。 デバッガ ウィンドウの既定のレイアウトは次のとおりです。 ● 既定では、[自動変数]、[ローカル]、および [ウォッチ 1] は IDE の一番下に表示されます。 ● [呼び出し履歴]、[ブレークポイント]、[スレッド]、[モジュール]、および [出力] は別のウィンドウ、または IDE の一番下に表示されます。 ● 出力ウィンドウは表示されません。

参照

処理手順

方法 : 選択した設定を変更する

方法 : チームの設定を指定する

関連項目

[設定のインポートとエクスポート] ([オプション] ダイアログ ボックス - [環境])

方法：Visual J# スタート キットを使用する

Visual J# 電卓スタート キットは、すぐに読み込んでビルドできる完全に独立したアプリケーションです。このスタート キットには、アプリケーションのカスタマイズに関する手順が記載された専用のドキュメントが付属しています。

Visual J# スタート キットを読み込み、ビルドするには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。

   [新しいプロジェクト] ダイアログ ボックスが表示されます。このダイアログ ボックスには、Visual J# で作成できる既定の各種アプリケーションが一覧表示されます。

2. [プロジェクトの種類] リストの [スタート キット] をクリックし、[テンプレート] ペインをクリックして [電卓 スタート キット] をクリックします。

3. [プロジェクト名] ボックスで、新規プロジェクトの名前を入力し、[OK] をクリックします。

   電卓スタート キット プロジェクトが開きます。

   ソリューション エクスプローラで、[ヘルプ] フォルダの Index.htm ファイルをクリックし、アプリケーションのビルドと拡張に関する、スタート キットの機能、アーキテクチャ、および手順の説明を確認します。

参照

その他の技術情報

Visual J# リファレンス

セキュリティに関する推奨事項 (Visual J#)

セキュリティは、アプリケーション配置の全段階で、十分に考慮する必要があります。以下のセクションでは、アプリケーションのセキュリティを高めるために役立ついくつかのヒントを示します。

セキュリティに関する Visual J# に固有の推奨事項

次の一覧は潜在的なセキュリティ問題のすべてを網羅してはいませんが、Visual J# の開発者にとって意識する必要がある共通の問題の一部を示しています。

- バッファ オーバーランは、大きなセキュリティ問題です。データをバッファにコピーする際は常に注意を払い、バッファ以外の場所に対する書き込みや読み取りをコードで実行できないようにする必要があります。チェックしていないユーザー入力をバッファにコピーしないでください。また、コピー対象データの範囲を決定するサイズ パラメータに未確認の値を使用しないでください。
- ファイル名に基づいて判断しないようにします。ファイル名はさまざまな方法で表現される可能性があり、特定のファイルに対するテストが省略される可能性があります。
- パスワードまたはその他の重要な情報はアプリケーション内にハード コーディングしないようにします。
- SQL クエリの生成に使用される入力は常に検証します。
- 例外情報を表示しないようにします。攻撃者に重要なヒントを与えてしまいます。
- 最低限の権限で実行されてもアプリケーションが機能することを確認します。管理者としてログインすることをユーザーに要求するアプリケーションはほとんどありません。
- XML またはその他の構成ファイルに重要な情報を格納しないようにします。
- アプリケーションの外部から渡されたデリゲートを使用する場合は、注意して使用します。
- アセンブリに対して FxCop を実行し、Microsoft .NET Framework 設計ガイドラインに準拠していることを確認します。また、FxCop では、200 種類を越えるコードの問題を検出して警告を発することもできます。

セキュリティに関する一般的な推奨事項

次に示す MSDN トピックでは、セキュリティで保護された信頼できるソフトウェアを作成するための詳細な情報について説明しています。

- Microsoft Security Developer Center
- Defend Your Apps and Critical User Info with Defensive Coding Techniques
- .NET Framework におけるセキュリティ概要
- Defend Your Code with Top Ten Security Tips Every Developer Must Know
- Security Brief: Beware of Fully Trusted Code
- 報告された Microsoft ASP.NET の脆弱性に関する情報
- Windows フォームのセキュリティの概要
- 安全なマネージ コントロールの作成

参照

その他の技術情報

Visual J# の概要
Visual J# リファレンス

初めて J# アプリケーションを作成する場合

ここでは、2 つの "Hello World" プログラムを作成します。1 つは、コンソール アプリケーションとして、もう 1 つは Windows アプリケーションとして作成します。

このセクションの内容

初めてのコンソール アプリケーション

簡単な Hello World コンソール アプリケーションの作成方法について説明します。

初めての Windows アプリケーション

簡単な Hello World Windows アプリケーションの作成方法について説明します。

参照

その他の技術情報

Visual J# リファレンス

初めての Windows アプリケーション

J# アプリケーションの作成に必要な時間はわずか 1 分です。次の手順に従うと、ウィンドウを開き、ボタンのクリックに応答するプログラムを数十秒で作成できます。

プロシージャ

Windows アプリケーションを作成するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。
2. Windows アプリケーション テンプレートが選択されていることを確認し、[プロジェクト名] フィールドに「MyWindowsProject」と入力します。[OK] をクリックして、プロジェクトを作成します。

   Windows フォーム デザイナに Windows フォームが表示されます。これがアプリケーションのユーザー インターフェイスです。
3. [表示] メニューの [ツールボックス] をクリックし、コントロールの一覧を表示します。
4. Label コントロールをフォーム中央にドラッグします。
5. ツールボックスから、フォーム上のラベルの近くにボタンをドラッグします。
6. ボタンをダブルクリックしてコード エディタを開きます。

   ボタンがクリックされたときに実行される **button1_Click** というメソッドが Visual J# によって挿入されていることがわかります。
7. このメソッドを次のように変更します。

```
private void button1_Click(object sender, EventArgs e)
{
    label1.set_Text("Hello, World!");
}
```

8. F5 キーを押して、アプリケーションをコンパイルおよび実行します。

   このボタンをクリックすると、"Hello, World!" というメッセージが表示されます。これで処理は完了しました。初めての J# アプリケーションが完成しました。

参照

その他の技術情報

Visual J# リファレンス

初めてのコンソール アプリケーション

コンソール アプリケーションは、IDE で簡単に作成できます。次の手順に従うとプログラムを数十秒で作成できます。また、コマンド ライン環境でアプリケーションを作成したりビルドしたりすることもできます。詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

プロシージャ

IDE 環境でアプリケーションをビルドするには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。
2. [コンソール アプリケーション] テンプレートを選択し、[プロジェクト名] フィールドに「**MyConsoleProject**」と入力します。
3. [OK] をクリックして、プロジェクトを作成します。コード エディタが表示されます。
4. メソッド `main` を見つけ、左中かっこの後に次の行を挿入します。

```
System.out.println("Hello World!");
```

5. [デバッグ] メニューの [デバッグなしで開始] をクリックするか、Ctrl キーを押しながら F5 キーを押します。[コンソール] ウィンドウが開き、"Hello World!" と表示されます。
6. 任意のキーを押すと、アプリケーションが終了します。

参照

その他の技術情報

Visual J# リファレンス

Visual J# での操作方法

「操作方法」のページには、Visual J# のプログラミングおよびアプリケーション開発に関連した主要なタスク ベースのトピックが紹介されています。Visual J# で行うことができる操作の基本的なカテゴリの一覧を以下に示します。該当するリンクをクリックすると、重要な手順について解説されたヘルプ ページに移動できます。

Windows アプリケーション (Visual J# での操作方法)

ActiveX コントロール、その他の各種コントロール、ダイアログ ボックス、ドラッグ アンド ドロップ、描画とグラフィックスなどについて説明します。

開発環境 (Visual J# での操作方法)

ナビゲーション、編集などについて説明します。

言語 (Visual J# での操作方法)

Java と J#、J# と .NET Framework、例外、リフレクションなどについて説明します。

インストールとセットアップ (Visual J# での操作方法)

インストール、構成などについて説明します。

コンソール アプリケーション (Visual J# での操作方法)

操作方法、J# アプリケーションのコンパイルなどについて説明します。

プロジェクト、ソリューション、ファイル (Visual J# での操作方法)

操作方法、プロジェクトとファイル、プロジェクトの種類、ソリューションなどについて説明します。

配置 (Visual J# での操作方法)

依存関係、Windows アプリケーション、Web アプリケーション、Web サービスなどについて説明します。

アップグレード (Visual J# での操作方法)

Visual J++ 6.0 からのアップグレード、新機能、複数バージョンの side-by-side 実行などについて説明します。

Web アプリケーション (Visual J# での操作方法)

Web サイト、Web コントロール、ASP.NET などについて説明します。

データ アクセス (Visual J# での操作方法)

データベースへの接続、データの表示、クエリ、データの編集などについて説明します。

コーディング タスク (Visual J# での操作方法)

ファイルと入出力、COM、リソース、シリアル化などについて説明します。

セキュリティ (Visual J# での操作方法)

ベスト プラクティス、Web セキュリティ、Windows セキュリティなどについて説明します。

XML Web サービス (Visual J# での操作方法)

作成、アクセス、検索などについて説明します。

デバッグ (Visual J# での操作方法)

Windows アプリケーション、Web アプリケーション、ストアド プロシージャなどについて説明します。

コマンド ライン (Visual J# での操作方法)

ツール、コンパイルなどについて説明します。

リファレンス (Visual J# での操作方法)

Visual J# 言語、Visual J# ライブラリなどについて説明します。

参照

その他の技術情報

Visual J#

コーディング タスク (Visual J# での操作方法)

ここでは、Visual J# のコーディングに関連するタスク全般について扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

全般

Visual J#

J# の概要のほか、J# について扱った各種トピックへのリンクを提供します。

ファイルと入出力

OpenRead

File.OpenRead メソッドを使用し、既存のファイルを読み取り用に開く方法について説明します。

OpenWrite

File.OpenWrite メソッドを使用し、既存のファイルを書き込み用に開く方法について説明します。

COM

デザインの概要 : COM コンポーネントのアクセス

COM オブジェクトは、Visual J++ 6.0 と同じ **com.ms.*** クラスとインターフェイスを継承するオブジェクトとして引き続き公開されます。この点について具体的に説明します。

方法 : COM コンポーネントを J# で生成する

COM コンポーネントの作成方法およびクライアントがコンポーネントを使用できるしくみについて説明します。

方法 : VJ++ で記述された COM コンポーネントにアクセスする

プロジェクトの COM コンポーネントへの参照を作成する方法について説明します。

方法 : COM からのマネージ コンポーネントへのアクセス

COM からマネージ コンポーネントにアクセスする手順について説明します。

OS へのアクセス

方法 : ソケットを作成する

コンストラクタのパラメータを使用し、プロトコル情報およびネットワーク情報でソケットを初期化する方法について説明します。

スレッドを作成し、開始時にデータを渡す

Thread オブジェクトの新しいインスタンスを作成することによって、新しいマネージ スレッドを作成する方法について説明します。

マルチスレッド処理のためのデータの同期

複数のスレッドが同じオブジェクトのプロパティとメソッドを呼び出す場合は、これらの呼び出しを同期することが重要です。この点について具体的に説明します。

リソース

Visual J# アプリケーションでのリソースの使用

エラー メッセージなどのリソース ファイル (.resx) を Visual J# プロジェクト内に格納する方法について説明します。

方法 : .properties ファイルからリソースにアクセスする

従来の .properties ファイルに対する読み書きの方法を示す例が紹介されています。

方法 : リソースを使う Visual J++ 6.0 アプリケーションのアップグレード

Visual J++ 6.0 プロジェクト リソースを手動で .NET Framework リソースに変換する手順について説明します。

Vjsresgen サンプル (Visual J++ 6.0 リソース ファイルの .NET Framework リソース ファイル (.resx) への変換)

Visual J++ 6.0 のリソース ファイルを .NET Framework のリソース ファイル (.resx) に変換する方法について説明します。

シリアル化

方法 : J# クラスをバイナリとしてシリアル化する

バイナリ シリアル化の使用方法を示した例が紹介されています。

方法 : Java アプリケーションで .NET Framework シリアル化を有効にする

J# アプリケーション データをシリアル化したり格納したりする方法の例が紹介されています。

.NET シリアル化のサポート

Serializable を実装するオブジェクトを作成し、J# クラス ライブラリの型をフィールドとして使用する例が紹介されています。

方法 : .NET Framework を使用して J# クラスのカスタム シリアライザを記述する

カスタムのシリアル化処理を使用して、GPS 座標を保存したり取得したりする方法を紹介します。

方法 : .NET Framework から、シリアル化された Java オブジェクトを逆シリアル化する

基本顧客情報をシリアル化および逆シリアル化することによってアプリケーション情報をキャッシュする例が紹介されています。

方法 : Java シリアル化オブジェクト (.ser) を .NET Framework シリアル化オブジェクトに変換する

Java のシリアル化アプリケーションを、.NET Framework のシリアル化オブジェクトを使って Visual J# アプリケーションに変換する方法の例が紹介されています。

データ アクセス

方法 : Windows フォーム DataGridView コントロールにデータを表示する

DataGridView コントロールを使用し、Windows フォーム上にデータを表示する方法について説明します。

方法 : 既存のコントロールにデータをバインドする

既存の Windows フォーム コントロールを、データ ソース内のデータにバインディングする方法について説明します。

方法 : Windows フォームの ComboBox または ListBox コントロールをデータにバインドする

ComboBox コントロールまたは **ListBox** コントロールのバインディングについて説明します。

チュートリアル : データセットへの XML データの読み込み

XML データをデータセットに読み込む Windows アプリケーションの作成手順について説明します。その後、データセットを **DataGridView** に表示します。最後に、XML ファイルの内容に基づいた XML スキーマがテキスト ボックスに表示されます。

方法 : アプリケーションで JDBC を使用する

データ ソースの名前とデータのアクセス方法をセットアップする例が紹介されています。

データに関するチュートリアル

データおよび XML に関する一般的な事例を、一連のトピックを通じて段階的に説明します。

ASP.NET データ アクセス (Visual Studio)

データ ドリブンの Web ページ作成について説明します。

その他

方法 : クラス ローダーを記述する

既存の Java 言語アプリケーションのカスタム クラス ローダーを変更して、マネージ アセンブリからクラスを読み込む方法を示した例が紹介されています。

OpenRemoteBaseKey

リモート コンピュータのレジストリ キーを開いて、キーの値を列挙するコード例が紹介されています。リモート コンピュータで、リモート レジストリ サービスが実行されている必要があります。

Random

パラメータなしのクラス コンストラクタでランダム オブジェクトを作成し、一連の整数と倍精度浮動小数点数を無作為に生成するコード例が紹介されています。

参照

関連項目

Visual J# での操作方法

コマンド ライン (Visual J# での操作方法)

ここでは、Visual J# のコマンド ラインに関連するタスク全般について扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

全般
Visual J# コンパイラ

J# コンパイラについて説明します。

ツール
コマンド ラインからのビルド (Visual J#)

コマンド ラインを使用したビルドについて説明します。

Visual J# コンパイラ オプション一覧 (カテゴリ別)

コマンド ライン コンパイラ オプションの使用方法について説明します。

コンパイル
J# アプリケーションのコンパイル

実行可能ファイル名 (vjc.exe) をコマンド ラインに入力することによって Visual J# コンパイラを呼び出す方法について説明します。

参照
関連項目
Visual J# での操作方法

コンソール アプリケーション (Visual J# での操作方法)

ここでは、Visual J# のコンソール アプリケーションに関するタスク全般を扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

方法 : コンソール アプリケーションへのユーザー入力を検証する

パラメータを使用して、プログラムの動作を制御する方法について説明します。

方法 : コンソール アプリケーション クライアントを作成する

XML Web サービスのコンソール クライアント アプリケーションを作成する方法について説明します。

コマンド ラインからのビルド (Visual J#)

実行可能ファイル名 (vjc.exe) をコマンド ラインに入力することによって Visual J# コンパイラを呼び出す方法について説明します。

J# アプリケーションのコンパイル

J# のソース コードを実行可能ファイル (.exe) またはダイナミック リンク ライブラリ (dll) アセンブリにコンパイルする方法について説明します。コンパイル後のバイナリ アセンブリは、コンパイラのスイッチ設定に基づいて bin\debug または bin\release ディレクトリに存在します。

参照

関連項目

Visual J# での操作方法

データ アクセス (Visual J# での操作方法)

ここでは、Visual J# のデータ アクセスに関連するタスク全般について扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

データ アクセス

ADO.NET、トランザクション処理、および XML を使用してデータにアクセスする方法について説明します。

データベースへの接続

方法 : SQL Server データベースへの接続を作成する

.NET Framework SQL Server 用データ プロバイダを使用して、アプリケーションを SQL Server に接続する方法について説明します。

方法 : Access データベースへの接続を作成する

Microsoft Jet 4.0 OLE DB プロバイダを使用して、Access データベースに接続する方法について説明します。

方法 : Oracle データベースへの接続を作成する

.NET Framework Oracle 用データ プロバイダを使用して、アプリケーションを Oracle データベースに接続する方法について説明します。

ConnectionString

System.Data.Odbc 名前空間を使用して ODBC データ ソースに接続する方法について説明します。

データ表示

方法 : 既存のコントロールにデータをバインドする

既存の Windows フォーム コントロールを、データ ソース内のデータにバインディングする方法について説明します。

方法 : Windows フォームの ComboBox または ListBox コントロールをデータにバインドする

ComboBox コントロールまたは **ListBox** コントロールのバインディングについて説明します。

クエリ

データセットへの読み込みとデータのクエリの概要

TableAdapter またはコマンド オブジェクトを使用し、データ ソースに対して SQL ステートメントまたはストアド プロシージャを実行する方法について説明します。

データの編集

DataView を使用したデータの並べ替えとフィルタ処理

DataView を使用した、DataTable 内データの並べ替えとフィルタ処理について説明します。

DataTable 内のデータの操作

DataSet 内に **DataTable** を作成した後で、データベース内のテーブルを使用する場合と同じ操作を実行できます。この点について具体的に説明します。テーブル内のデータの追加、表示、編集、および削除を実行したり、エラーとイベントを監視したり、テーブル内のデータを照会したりできます。

テーブルへのデータの追加

DataTable を作成し、列と制約を使用してそのテーブルの構造を定義した後で、テーブルに新しいデータ行を追加できます。この点について具体的に説明します。

テーブルからの行の削除

データ テーブルからレコードを削除する方法について説明します。

テーブル内のデータの表示

DataTable を使用し、指定した検索条件に基づいて、データのサブセットを表示する方法について説明します。

DataTable 内のデータの操作

DataSet 内に **DataTable** を作成した後で、データベース内のテーブルを使用する場合と同じ操作を実行できます。この点について具体的に説明します。テーブル内のデータの追加、表示、編集、および削除を実行したり、エラーとイベントを監視したり、テーブル内のデータを照会したりできます。

方法 : DataTable の行を編集する

DataTable で既存の行を編集するには、編集する DataRow を検索し、更新した値を目的の列に割り当てる必要があります。この点について具体的に説明します。

ストアド プロシージャ
ストアド プロシージャのサンプル

ストアド プロシージャの呼び出し方法について説明します。

方法 : ストアド プロシージャおよびユーザー定義関数を作成する

サーバー エクスプローラを使用して、ストアド プロシージャを作成する方法について説明します。

参照
関連項目
Visual J# での操作方法

デバッグ (Visual J# での操作方法)

ここでは、Visual J# のデバッグに関連するタスク全般について扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

チュートリアル : Windows フォームのデバッグ

最も一般的なマネージ アプリケーションの 1 つである Windows フォームのデバッグ方法について説明します。

Web アプリケーションのデバッグ

スクリプトや Web アプリケーションのデバッグ時に発生する一般的な問題、およびデバッグの手法を扱ったトピックへのリンクが掲載されています。

Windows アプリケーション
チュートリアル : デザイン時のデバッグ

デバッグするコードの準備とブレークポイントの設定方法について説明します。

方法 : デバッガを自動的に起動する

Windows でアプリケーションを起動したときに Visual Studio が起動されるようにアプリケーションを設定する方法について説明します。

デバッグの準備 : Windows フォーム アプリケーション

Windows アプリケーションのデバッグ構成について説明します。

Web アプリケーション
デバッグの準備 : Web アプリケーション

Web サイト プロジェクト テンプレートで作成された基本的なプロジェクトの種類をデバッグする方法について説明します。

デバッグの準備 : ASP.NET Web アプリケーション

Web アプリケーションのデバッグ構成について説明します。

デバッグの準備 : XML Web サービス プロジェクト

XML Web サービス アプリケーションのデバッグ構成について説明します。

ストアド プロシージャ
方法 : ストアド プロシージャおよびユーザー定義関数を作成する

ストアド プロシージャが、複雑なビジネス ルールの定義、データ変更の制御、セキュリティ設定によるアクセス制限、トランザクションの整合性のほか、アプリケーションで必要なデータベース処理をどのように実現するかについて説明します。

方法 : ストアド プロシージャおよびユーザー定義関数を変更する

ストアド プロシージャおよびユーザー定義関数に対してクエリ デザイナを使用し、SQL テキストのブロックや選択したテキストを変更したり、新しい SQL テキストを挿入する方法について説明します。

参照
関連項目
Visual J# での操作方法

配置 (Visual J# での操作方法)

ここでは、Visual J# の配置に関連するタスク全般について扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

全般
Visual J# アプリケーションの配置

XML Web サービスまたは Web フォーム アプリケーションを配置するサーバーに何をインストールしておく必要があるかについて説明します。

依存関係
.NET Framework の再頒布

.NET Framework 用に作成されたアプリケーションおよびコントロールでは、アプリケーションやコントロールを実行するコンピュータ上に .NET Framework をインストールする必要があります。この点について具体的に説明します。

Visual J# 再頒布可能パッケージの配布

Visual J# で記述されたアプリケーションやコントロールを使うには、そのアプリケーションまたはコントロールを実行するコンピュータに Visual J# .NET 再頒布可能パッケージをインストールする必要があります。この点について具体的に説明します。

Windows アプリケーション
チュートリアル : Windows ベースのアプリケーションの配置

Windows アプリケーションを配置するために必要な作業について説明します。

Web アプリケーション
チュートリアル : XCOPY を使用した ASP.NET Web アプリケーションの配置

ASP.NET Web アプリケーションを配置するために必要な手順について説明します。

方法 : J# ブラウザ コントロールを配置する

単純なコントロールを作成し、そのコントロールを Web サイトに配置し、Web ページ上に表示する方法を示した例が紹介されています。

配置方法の選択
Visual Web Developer での Web サイトの配置

Web サイトを開発用コンピュータから、ステージングまたは実行用の Web サーバーにコピーする際の選択肢について説明します。

チュートリアルの一覧の配置

一般的な配置シナリオの例を段階的に説明するトピックの一覧が紹介されています。

アプリケーションまたはコンポーネントの複数バージョンの配置

利用可能な配置の種類について説明します。

Web サービス
XML Web サービスの配置

Web サービスを配置するときには、Web サービスによって使用されるが、Microsoft .NET Framework には付属していない .asmx ファイルおよびアセンブリを Web サーバーにコピーする必要があります。この点について具体的に説明します。

XML Web サービス探索

発行した .disco ファイルが、どのようにして、プログラムによる XML Web サービス検索を実現しているかについて説明します。

コンポーネント
方法 : マージ モジュール プロジェクトの作成または登録を行う

マージ モジュールの作成、および、マージ モジュールのプロジェクトへの追加について説明します。

参照
関連項目
Visual J# での操作方法

開発環境 (Visual J# での操作方法)

ここでは、Visual J# の開発環境に関連するタスク全般について扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

全般

Visual J# の概要

Visual J# 開発システムの機能、特徴、およびアーキテクチャの概要を説明します。

ナビゲーション

ウィンドウの管理

統合開発環境 (IDE) でのウィンドウ管理を支援するために Visual Studio が備えたさまざまなツールとオプションについて説明します。

Visual Studio の統合開発環境

開発プロセスにおける Visual Studio の使用方法を紹介したトピックへのリンクを提供します。

[編集]

方法 : コードをアウトライン表示する/非表示にする

アウトラインの中止コマンドを使用し、元のコードに影響を及ぼすことなく、アウトライン表示を中止する方法について説明します。

[全般] ([オプション] ダイアログ ボックス - [環境])

[オプション] ダイアログ ボックスでオートコンプリートをオフにする方法について説明します。

IntelliSense オプションの変更

IntelliSense オプションを既定でオフにして、メニュー コマンドまたはキーストロークを組み合わせて呼び出す方法について説明します。

エディタの便利なコマンドおよび機能

コードの入力時またはファイルを開いたときに表示される、構文エラーを示す波線について説明します。この波線は、エラーを修正するとすぐに消えます。

方法 : クラス ビューを使用する

クラス ビューを使用してクラス定義を表示する方法について説明します。

方法 : カスタム コメント トークンを作成する

開発言語に固有のコメント マークで始まるコード内コメントについて説明します。コメント マークの後にトークン文字列 (TODO、UNDONE、HACK、またはカスタム トークン文字列) を挿入すると、対応するコメントへのショートカットがタスク一覧に表示されます。

方法 : Javadoc コメントから XML ドキュメントを生成する

ファイルまたはプロジェクト内のコメントを、Java 言語のコメント形式から .NET Framework 標準のコメント形式に変換する方法について説明します。

参照

関連項目

Visual J# での操作方法

インストールとセットアップ (Visual J# での操作方法)

ここでは、Visual J# のインストール タスクおよびセットアップ タスク全般について扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

インストール

ユーザーのアクセス許可および高度な設定のオプション

Visual J# のインストールについて説明します。

製品のサポートとユーザー補助

Visual J# インストールの修復について説明します。

方法 : Visual Studio のヘルプをインストールする

Visual Studio ヘルプのセットアップと構成に関する情報およびヒントを提供します。

ユーザーのアクセス許可および高度な設定のオプション

複数の Visual Studio 製品を共存させるために必要な情報が記載されています。

ユーザーのアクセス許可および高度な設定のオプション

インストールに関するヘルプを表示する方法について説明します。

製品のサポートとユーザー補助

インストール エラーのトラブルシューティングについて説明します。

ユーザーのアクセス許可および高度な設定のオプション

インストールで選択可能なオプションについて説明します。

参照

関連項目

Visual J# での操作方法

言語 (Visual J# での操作方法)

このページでは、よく使用する J# 言語タスクに関するヘルプへのリンクを紹介します。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

Visual J# の概要

J# を初めて使用するユーザーのための情報が記載されています。

Java と J#

方法 : Try ブロックと Catch ブロックを使用して例外をキャッチする

try-catch ブロックを使用して例外をキャッチするコード例が紹介されています。

方法 : 列挙定数を作成する

enum キーワードを使って独自のデータ型を作成する方法について説明します。これらのデータ型を使用して、変数への割り当てが可能なすべての値を定義する名前、またはその他のリテラル値のセットを宣言できます。

方法 : package スコープのクラスを記述する

package ステートメントを使用して、アプリケーション内の関連するクラスをグループ化する方法について説明します。同じパッケージ名を持つクラスは package スコープのクラス と呼ばれます。スコープを設定すると、同じパッケージ内のクラス メンバを相互に使用できます。また、クラスの呼び出し元は、簡単な import ステートメントを使用してスコープ全体にアクセスできます。

方法 : Java 言語バイトコードを MSIL コードに変換する

さまざまなレガシ Java アプリケーションを Visual J# に容易に変換する方法について説明します。Visual J# バイナリ変換ツール (JbImp.exe) は、Java 言語のバイトコード (.class) ファイルを Microsoft Intermediate Language (**MSIL**) に変換します。

Supplemental UI Library

Microsoft Windows の外観と操作性を備えた、アプレットをはじめとする各種のアプリケーションを開発する方法について説明します。

J# と .NET

方法 : J# にインターフェイスを実装する

インターフェイスと同じシグネチャを持つメソッドを宣言し、それらを実装する方法について説明します。

属性の追加

Visual J# を使用して、クラス、フィールド、メソッド、パラメータ、およびその他のプログラミング要素に属性を追加する方法について説明します。

参照渡しの引数を受け取るメソッドの呼び出し

メソッドを呼び出し、そのパラメータを参照渡しで渡す方法について説明します。

プロパティの定義と使用

Visual J# を使用して、.NET Framework のクラスで定義されているプロパティにアクセスする方法について説明します。

イベントの定義と使用

.NET Framework クラスにより発行されたイベントを登録する方法について説明します。

列挙型の使用

.NET Framework またはその他のユーザー定義アセンブリで定義された列挙型を J# コンパイラで使用する方法について説明します。

共通言語仕様への準拠の検証

作成したコードが CLS に準拠しているかどうかを検証する方法について説明します。CLS 準拠とは、すべての言語に共有されていない言語構成要素の使用を避けることを意味します。CLS 準拠でない場合、その型をすべての言語で使用できるとは限りません。

例外

例外処理の基本事項

例外処理に関する一連の推奨事項が紹介されています。

例外の処理とスロー

.NET Framework および Visual Studio を使用した例外処理の概要についてベスト プラクティスを含めて説明します。

方法 : Try ブロックと Catch ブロックを使用して例外をキャッチする

try-catch ブロックを使用して例外をキャッチするコード例が紹介されています。

方法 : catch ブロックで特定の例外を使用する

特定の例外を対象とした **catch** ブロックを、一般的な例外の catch ブロックの前に配置する方法について説明します。

リフレクション
リフレクションの概要

リフレクションの概念と、その使用方法を解説したトピックへのリンクを提供します。

ネイティブ関数へのアクセス
ネイティブ メソッドの呼び出し

共通言語ランタイムに用意されているプラットフォーム呼び出しサービスを使用して、ネイティブ コード関数を呼び出す方法について説明します。

方法 : Visual J# から DLL のメソッドを呼び出す

vjslib.dll ライブラリから静的メソッドおよび動的メソッドを呼び出す方法の例が紹介されています。

PInvoke サンプル (アンマネージ API の呼び出し)

共通言語ランタイムに用意されているプラットフォーム呼び出しサービスを使って、Visual J# でアンマネージ API を呼び出す方法について説明します。

標準と規則
方法 : Java 言語アプリケーションを Visual J# でコンパイルする

独自のプロジェクトを作成した後、作成したプロジェクトに既存の .java ファイルを追加して J# でコンパイルする方法について説明します。

共通言語仕様への準拠の検証

型が共通言語仕様 (CLS) に準拠しているかどうかをチェックする方法について説明します。

参照
関連項目
Visual J# での操作方法

プロジェクト、ソリューション、ファイル (Visual J# での操作方法)

このページでは、よく使用する J# プロジェクト タスクに関するヘルプへのリンクを紹介します。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

方法 : セットアップ プロジェクトを作成または登録する

2 種類のセットアップ プロジェクトの作成方法について説明します。通常のセットアップ プロジェクトは、Windows アプリケーションをターゲット コンピュータにインストールするためのインストーラを作成します。Web セットアップ プロジェクトでは、Web アプリケーションを Web サーバーにインストールするためのインストーラを作成します。

方法 : アセンブリ情報を指定する

プロジェクト デザイナの [アプリケーション] ペインからアクセスできる [アセンブリ情報] ダイアログ ボックスに対し、アセンブリ情報を入力する方法について説明します。

方法 : プロジェクトによるファイルの管理方法を識別する

プロジェクト ファイルの格納場所とそれらの管理方法について説明します。

方法 : ビルドのプロパティを設定する (C#、J#)

Visual Studio プロジェクト デザイナの [ビルド] ペインと [ビルド イベント] ペインを使用して、ビルド設定およびイベントを指定する方法について説明します。

方法 : 複数のプロジェクトから成るソリューションを作成する

共通のソリューション ディレクトリに複数のプロジェクトを作成することによって、ソース コード管理下の多数の関連プロジェクトを共有する方法について説明します。

プロジェクトとファイル

How to: Add Files to a Project

Application Builder または Project Manager を使用して、既存のファイルをプロジェクトに追加する方法について説明します。

方法 : ファイルまたはプロジェクトの名前を変更する

プロジェクトの各要素の名前を変更する方法について説明します。

方法 : エンコーディングでファイルを管理する

エンコーディングを使用してファイルを管理する方法について説明します。作成したコードを特定の言語および特定のプラットフォームで簡単に表示できるようにするには、ファイルに特定の文字エンコーディングを関連付けます。

How to: Remove Files from Projects

ファイルをプロジェクトから除外したり、Project Manager を使って削除したりする方法について説明します。

方法 : 既存のコード ファイルからプロジェクトを作成する

Visual Studio 統合開発環境で既存のコード ファイルを新しいプロジェクトで使用する方法について説明します。

Project Types and Items

プロジェクトの種類を指定し、対象のプロジェクトに属するファイルを、プロジェクト固有のエディタで開く方法について説明します。

プロジェクトの種類

方法 : コンソール アプリケーション クライアントを作成する

XML Web サービスのコンソール クライアント アプリケーションを作成する方法について説明します。

方法 : 新しい Swing アプリケーションを作成する

Swing のさまざまな機能を紹介するほか、**AWT** ライブラリの使用方法について説明しています。

方法 : ブラウザで実行されるコンポーネントを作成する

ライブラリ コンポーネントを作成し、そのライブラリ コンポーネントを Web ページに追加し、Web サイトにコンポーネントを配置する方法について説明します。

ソリューション

方法 : 複数のプロジェクトから成るソリューションを作成する

新しいプロジェクトを追加する方法と、既存のプロジェクトをソリューションに追加する方法について説明します。

参照

方法 : リモート Web 参照を追加および削除する

Web 参照を追加したり削除したりする方法について、コード例を交えて説明します。

[参照の追加] ダイアログ ボックス

[参照の追加] ダイアログ ボックスについて説明します。

壊れた参照のトラブルシューティング

アプリケーションが、無効な参照を利用しようとした場合に発生するエラーを、どのようにして修正すればよいかについて説明します。

Web 参照

プロジェクトの Web 参照は、目的に合わせて名前を変更したり修正を加えたりするか、またはその参照が呼び出す Web サービスに変更が加えられたときに更新することによって管理します。この点について具体的に説明します。

方法 : Visual Studio のリファレンスを追加または削除する

アプリケーションに、あらかじめコンポーネントの参照を追加する方法について説明します。

方法 : COM オブジェクトを参照する

[参照の追加] ダイアログ ボックスを使用して COM オブジェクトを参照する方法について説明します。

ビルドとコンパイル

方法 : アプリケーションのデバッグのための開始オプションを設定する

プロジェクトが実行モードになるときの動作を追加指定する方法について説明します。

Visual Studio でのビルド

既定のソリューション構成とプロジェクト構成を必要に応じて編集したり、変更された既定の構成のコピーを保存して新規の構成を作成したりする方法について説明します。

方法 : プロジェクトのプロパティを設定する (C#、J#)

プロジェクトのプロパティを使用して、プロジェクトのビルド方法とデバッグ方法を指定します。

テスト

方法 : オブジェクトのインスタンスを作成して表示する

クラス デザイナまたはクラス ビューからオブジェクトのインスタンスを作成する方法について説明します。

方法 : 返されたオブジェクトのインスタンスを表示する

取得したオブジェクトのインスタンスを、オブジェクト テスト ベンチに図で表示する方法について説明します。

方法 : オブジェクト インスタンスを調べて変更する

オブジェクトのビジュアルな表現を可能にするオブジェクト テスト ベンチの使用方法について説明します。

方法 : オブジェクト テスト ベンチからメソッドを呼び出す

オブジェクト テスト ベンチに表示されたオブジェクト図からメソッドを呼び出す方法について説明します。

リソース

チュートリアル : Windows フォームのローカリゼーション

Windows フォーム アプリケーションのローカライズ用に Visual Studio がサポートしている機能の使用方法について説明します。

アプリケーションのリソース

リソース ファイル内のデータを並べ替え、アプリケーション全体を再コンパイルすることなくデータだけを変更する方法について説明します。

リソース ファイルの操作

新しいリソースをプロジェクトに追加したり、これらのリソースを適切なリソース エディタを使用して変更したりする方法について説明します。

参照

関連項目

Visual J# での操作方法

リファレンス (Visual J# での操作方法)

このページでは、よく使用する J# プログラミング タスクに関するヘルプへのリンクを紹介します。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

Visual J# リファレンス

Visual J# プログラミングに関するさまざまなリファレンスへのリンクがあります。

Visual J# 言語
.NET Framework を対象とするための構文

.NET Framework を対象とするために使用される Visual J# コンパイラの機能について説明します。

Visual J# ライブラリ
クラス ライブラリのサポート

Visual J# でサポートされているパッケージとサポートされていないパッケージについて説明します。また、クラス ライブラリのサポートに関する実装固有の詳細情報についても説明します。

参照
関連項目
Visual J# での操作方法

セキュリティ (Visual J# での操作方法)

このページでは、よく使用する J# セキュリティ タスクに関するヘルプへのリンクを紹介します。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

コードのセキュリティとコンポーネントの署名

コードが配置されたときに常にそのコードが実行されるようにする方法と、コンポーネントを不正使用から保護する方法について説明します。

ProtectedMemory.Protect Method

データ保護の使用方法を示した例が紹介されています。

Visual J# バイナリ変換ツール (Java 言語のバイトコードから MSIL への変換用)

Java 言語の **バイトコード** (.class) ファイルを Microsoft® Intermediate Language (MSIL) に変換する Visual J# バイナリ変換ツール (JbImp.exe) について説明します。

最適な使用方法
アプリケーションの保護

アプリケーションを保護する上での課題について説明します。

Windows のセキュリティ
Windows フォームのセキュリティ

Windows フォームのコード ベースのセキュリティ モデル (コードを実行するユーザーに関係なく、セキュリティ レベルがコードに対して設定されるセキュリティ モデル) について説明します。これは、コンピュータ システム上に既に用意されている任意のセキュリティ スキーマに追加して使用されます。

Secure Printing In Windows Forms

印刷に関連した各種のアクセス許可レベルで利用できる機能について説明します。

Windows フォームにおけるファイルおよびデータへのより安全なアクセス

信頼度の低い環境で動作しているアプリケーションからファイル、データベース、およびレジストリにアクセスする方法について説明します。

Web セキュリティ
ASP.NET を使用して作成した XML Web サービスのセキュリティ

ASP.NET がさまざまな認証オプションおよび承認オプションを提供するためにインターネット インフォメーション サービス (IIS: Internet Information Services) とどのように連動するかを説明します。

Web アプリケーションの実行時のセキュリティ

一連の機能を利用して、アプリケーションを不正アクセスから保護する方法について説明します。

参照
関連項目
Visual J# での操作方法

アップグレード (Visual J# での操作方法)

このページでは、よく使用する J# アップグレード タスクに関するヘルプへのリンクを紹介します。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

Visual J++ ユーザーへの Visual J# の紹介

Visual J# によって .NET Framework に Java 言語のサポートが追加される方法について説明します。

Visual J++ 6.0 からのアップグレード

方法 : リソースを使う Visual J++ 6.0 アプリケーションのアップグレード

既存の Visual J++ リソースをマネージ アセンブリに埋め込んだり、リンクしたりすることによって共通言語ランタイムで使用するため、リソース ファイルをあらかじめ .NET Framework がサポートする形式に変換する方法について説明します。

方法 : Visual J++ プロジェクトを変換する

独自の Visual J++ プロジェクトを開いて変換する方法について説明します。

方法 : 既存の J++ コードから Visual J# プロジェクトを作成する

Visual J++ プロジェクトを Visual J# にアップグレードする例が紹介されています。

Visual J# のアップグレード レポート

アップグレード プロセスに関する情報と、プロジェクトの実行前に対処する必要のある問題について取り上げます。

方法 : J++/ActiveX アプリケーションをアップグレードする

Visual Studio 変換ウィザードを使用して、J++ ActiveX プロジェクトを Visual J# プロジェクトに変換する方法について説明します。

COM コンポーネントにアクセスする Visual J++ 6.0 アプリケーションのアップグレード

COM コンポーネントにアクセスする Visual J++ 6.0 の Java 言語/COM アプリケーションを Visual J# でコンパイルして実行する方法について説明します。

新機能

Visual J# 2005 の新機能

このリリースの Visual J# で追加された新機能を紹介します。

複数バージョンの side-by-side 実行

方法 : Visual J# 再頒布パッケージのバージョン管理を設定する

Visual Studio プロジェクト内からバージョンを設定する方法について説明します。

方法 : アプリケーション構成ファイルを使用して対象とする .NET Framework のバージョンを指定する

アプリケーション構成ファイルを使用して、アプリケーションまたはコンポーネントがサポートする .NET Framework のバージョンを指定する方法について説明します。

参照

関連項目

Visual J# での操作方法

Web アプリケーション (Visual J# での操作方法)

ここでは、Web アプリケーションに関連するタスク全般について扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

Web サイト

はじめに - Web サイトの作成

さまざまな Web サイトの作成、Web サーバーと Visual Web Developer の使用、および Web サイトのテストについて概要を紹介します。

詳細説明 - Web サイトの作成

ASP.NET Web サイトの使用に関する、より詳細な情報を提供します。

チュートリアルのトピック - ASP.NET Web サイト

Visual Web Developer で ASP.NET Web サイトを作成するチュートリアルへのリンクが掲載されています。

Web コントロール

ASP.NET Web サーバー コントロール

ASP.NET Web サーバー コントロールとは何か、また、それらをどのように使用すればよいかについて扱ったトピックへのリンクが掲載されています。

ASP.NET Web サーバー コントロールの操作

Web サーバー コントロールを ASP.NET Web ページに追加する方法やその外観を変更する方法のほか、Web サーバー コントロールのプログラミング方法について扱ったトピックへのリンクが掲載されています。

個々の ASP.NET Web サーバー コントロール

Web サーバー コントロールごとの情報を扱ったトピックへのリンクが掲載されています。

ASP.NET ユーザー コントロール

ユーザー コントロールの動作と作成方法、および、ユーザー コントロールを ASP.NET Web ページに追加する方法について扱ったトピックへのリンクが掲載されています。

Web ページ

チュートリアル : Visual Web Developer での基本的な Web ページの作成

Visual Web Developer を使って簡単な Web ページを作成する方法を、新しいページの作成、コントロールの追加、コードの記述に関する基本的なテクニックを交えながら説明します。

チュートリアル : Visual Web Developer でのコードの分離を使用した基本的な Web ページの作成

Visual Web Developer を使って、コードが分離された ASP.NET ページを作成する方法について説明します。

チュートリアル : Visual Web Developer での Web ページのコード編集

エラーの修正、コードのリファクタリング、シンボルの名前変更、コード スニペットの挿入について説明します。

データ アクセス

ASP.NET データ アクセスの概要

データ ソース コントロールおよびデータ バインド コントロールについて説明します。

方法 : データ ソース コントロールを使用するときに接続文字列をセキュリティ保護する

Web.config ファイルの connectionStrings 構成セクションに接続文字列を格納する方法のほか、コマンド ライン .NET Framework ツールを使って接続文字列を暗号化し、セキュリティを強化する方法について説明します。

方法 : テンプレート コントロールにデータを連結する

テンプレートを使用して、コントロールをデータにバインドする方法について説明します。

セキュリティ

ASP.NET Web アプリケーションのセキュリティ

ASP.NET のセキュリティ インフラストラクチャに関する情報のほか、ASP.NET の認証、承認、プロセス偽装の各機能について説明します。

メンバシップを使用したユーザーの管理

Web アプリケーションにおけるユーザー情報の検証と管理を可能にする、ASP.NET のメンバシップについて説明します。

ロールを使用した承認の管理

承認管理を通じて、アプリケーションの各ユーザーからのアクセスを許可するリソースを指定する方法について説明します。

保護された構成を使用した構成情報の暗号化

プロテクト構成を使った構成情報の暗号化のほか、関連するトピックへのリンクが掲載されています。

配置

方法 : 公開する Web サイトを構成する

発行する Web サイトの構成手順について説明します。

チュートリアル : XCOPY を使用した ASP.NET Web アプリケーションの配置

ASP.NET Web アプリケーションの個々のファイルを、コマンド ラインから配置または更新する手順について説明します。

方法 : セキュリティ ロックダウン コンソールの項目を管理する

セキュリティ ロックダウン コンソールでアイテムを管理する手順について説明します。

方法 : ASP.NET アプリケーションを特定の ASP.NET バージョン用に構成する

.NET Framework の特定のバージョンをターゲットとして ASP.NET アプリケーションを構成する手順について説明します。

参照

関連項目

Visual J# での操作方法

Windows アプリケーション (Visual J# での操作方法)

ここでは、Visual J# の Windows アプリケーションに関するタスク全般を扱ったヘルプへのリンクを紹介しています。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

全般

方法 : AWT のフレームを使用する

AWT を使用し、Microsoft® のメモ帳スタイルのエディタ アプリケーションを独自に作成する方法について説明します。

WFCUsingWinForm サンプル (Visual J++ 6.0 で作成した WFC アプリケーションの Windows フォームによる拡張)

Windows フォームを使って、Visual J++ 6.0 で作成した WFC アプリケーションを拡張する方法を示します。Visual J++ 6.0 で作成した単純な WFC アプリケーションを Visual J# にアップグレードします。

ActiveX コントロール

方法 : Windows フォームに ActiveX コントロールを追加する

Windows フォームに ActiveX コントロールを配置する方法について説明します。

コントロール

Button コントロールの概要 (Windows フォーム)

イベント ハンドラとボタン コントロールのバインディングについて説明します。

方法 : ツール バー コントロールにボタンを追加する

デザイン時およびプログラミング時にボタンを追加する方法について説明します。

方法 : デザイナを使用して Windows フォームの Button コントロールを承認ボタンとして指定する

ボタン コントロールを AcceptButton (既定のボタン) にする方法について説明します。

方法 : デザイナを使用して Windows フォームの Button コントロールをキャンセル ボタンとして指定する

ボタン コントロールを CancelButton として指定する方法について説明します。

方法 : ツール バー ボタンのアイコンを定義する

ボタンのアイコンをデザイン時およびプログラミング時に指定する方法について説明します。

TextBox.Wrap Property

複数行テキスト ボックス内でテキスト コンテンツが折り返すことができるかどうかを示す値を取得または設定する方法について説明します。

方法 : Windows フォームの RichTextBox コントロールにおける書式属性の変更を確認する

書式属性の変更に応答する方法について説明します。

方法 : Windows フォームの RichTextBox コントロールにスクロール バーを表示する

RichTextBox コントロールの **Scrollbars** プロパティに使用される 7 つの値について説明します。

方法 : Windows フォームの RichTextBox コントロールを使用して Web スタイルのリンクを表示する

RichTextBox コントロールに Web ページへのリンクを設定する方法について説明します。

方法 : Windows フォームの RichTextBox コントロールにファイルを読み込む

LoadFile メソッドを呼び出して、プレーンテキスト、Unicode プレーンテキスト、リッチ テキスト形式 (RTF) の各ファイルを表示します。

方法 : Windows フォームの RichTextBox コントロールを使用してファイルを保存する

ファイルを保存するには、**SaveFile** メソッドを呼び出します。

方法 : Windows フォームの RichTextBox コントロールのフォント属性を設定する

SelectionFont および **SelectionColor** の使用について説明します。

方法 : Windows フォームの RichTextBox コントロールを使用してインデント、ぶら下げインデント、および箇条書き段落を設定する

段落にインデントを設定したり、段落を箇条書きリストとして書式設定したりするためのプロパティの使用について説明します。

方法 : Windows フォーム CheckBox のクリックに応答する

チェック ボックスの状態に応じてアクションを実行するように、アプリケーションをプログラミングする方法について説明します。

方法 : Windows フォームの CheckBox コントロールでオプションを設定する

CheckBox コントロールを使用して、ユーザーが True/False または Yes/No のオプションを選択できるようにする方法について説明します。

方法 : Windows フォーム ComboBox、ListBox、または CheckedListBox コントロールの特定の項目にアクセスする

Windows Forms のコンボ ボックス、リスト ボックス、またはチェックされたリスト ボックスに表示される特定の項目にアクセスし、リストにどのような項目が含まれるかをプログラムから確認する方法について説明します。

方法 : Windows フォームの ComboBox、ListBox、または CheckedListBox コントロールに項目を追加または削除する

データ バインディングを使わずに、項目とコントロールをバインディングする最も簡単な方法を紹介します。

方法 : Windows フォーム ComboBox、ListBox、または CheckedListBox コントロールのルックアップ テーブルを作成する

コントロールにバインドするルックアップ テーブルの作成方法について説明します。

方法 : Windows フォームの ComboBox または ListBox コントロールをデータにバインドする

コントロールに対するデータのバインディングについて説明します。

方法 : Windows フォーム ComboBox、ListBox、または CheckedListBox コントロールを並べ替える

並べ替えがサポートされているデータ ソースを使用し、そのデータ ソースで並べ替えを行うことによって、並べ替えたデータを表示する方法について説明します。

方法 : Windows フォームの MonthCalendar コントロールの外観を変更する

Windows フォームの **MonthCalendar** コントロールの外観をカスタマイズする方法について説明します。

方法 : Windows フォームの MonthCalendar コントロールにおいて複数の月を表示する

コントロールに複数の月を表示し、それらを整列させる方法について説明します。

方法 : Windows フォームの MonthCalendar コントロールを使用して特定の日付を太字で表示する

個々の日付の外観を制御する 3 つのプロパティの使用方法について説明します。

方法 : Windows フォームの MonthCalendar コントロールで日付の範囲を選択する

月間予定表コントロールで日付範囲を選択する方法について説明します。

方法 : 実行時にピクチャのサイズまたは配置を変更する (Windows フォーム)

Windows フォームの **PictureBox** コントロールに対し、**SizeMode** プロパティを設定し、画像を整列させたり、伸縮させたりする方法について説明します。

方法 : デザイナを使用してピクチャを読み込む (Windows フォーム)

有効な画像を **Image** プロパティに設定することにより、デザイン時に画像を読み込んでフォームに表示する方法について説明します。

方法 : 実行時にピクチャを設定する (Windows フォーム)

Windows フォームの **PictureBox** コントロールで表示されるイメージをプログラムで設定する方法について説明します。

方法 : タブ ページにコントロールを追加する

タブ コントロールをプログラムによって追加する手順が紹介されています。

方法 : Windows フォーム TabControl のタブを追加および削除する

Windows フォームの **TabControl** は、デザイナを使用するか、またはコードによって追加したり削除したりできます。

方法 : セットとして機能する Windows フォーム RadioButton コントロールをグループ化する

独立した機能のセットとして **RadioButton** コントロールをグループ化する方法について説明します。

方法 : コントロールに透明な背景を指定する

透明な背景色は、コントロールの既定ではサポートされていません。ただし、コンストラクタで **Control.SetStyle** メソッドを使用すると、コントロールの背景色を不透明、透明、または半透明に設定できます。この点について具体的に説明します。

方法 : コントロールのコレクションに対して実行時にコントロールを追加または削除する

アプリケーション開発の一般的なタスクとして、フォーム上の任意のコンテナ コントロールにコントロールを追加したり、コンテナからコントロールを削除したりする方法について説明します。

チュートリアル : Visual J# でのユーザー コントロールの作成

簡単なユーザー コントロールを作成し、継承を使用して、このコントロールの機能を拡張する方法について説明します。

ダイアログ ボックス

方法 : Windows フォームのダイアログ ボックスを表示する

ダイアログ ボックスは、アプリケーションで表示されるその他のフォームと同じ方法で表示されます。

方法 : ダイアログ ボックスを閉じて、ユーザー入力を保持する

ダイアログ ボックスのすべての **Button** コントロールに対し、**DialogResult** プロパティをデザイン時に設定する方法について説明します。実行時に **DialogResult** プロパティを設定すると、ユーザーの応答を動的に処理できます。

方法 : デザイン時にダイアログ ボックスを作成する

デザイン時にダイアログ ボックスを作成する方法について説明します。

ドラッグ アンド ドロップ

方法 : Windows フォームの RichTextBox コントロールにおけるドラッグ アンド ドロップ操作を有効にする

DragEnter イベントと **DragDrop** イベントを処理することによってドラッグ操作を実行する方法について説明します。

描画とグラフィックス

Width

ペンの幅を取得したり設定したりする方法について説明します。

方法 : デザイナを使用してピクチャを読み込む (Windows フォーム)

有効な画像を **Image** プロパティに設定することにより、デザイン時に画像を読み込んでフォームに表示する方法について説明します。

直線、曲線、および図形

曲線を含むさまざまな線や図形を描画する方法について説明します。

イベント処理

方法 : Windows フォームで実行時にイベント ハンドラを作成する

実行時にイベント ハンドラを作成することにより、プログラムが起動した時点でイベント ハンドラを接続するのではなく、実行時にコード内の条件に基づいてイベント ハンドラを接続できます。この点について具体的に説明します。

フォーム

方法 : Windows フォームの画面上の位置を設定する

Location プロパティに値を入力することによって、コンピュータ画面上のフォームの表示位置を指定する方法について説明します。

方法 : 四角形以外の Windows フォームを作成する

フォームの標準的な形状をカスタマイズする方法について説明します。

方法 : Windows フォームを常に一番手前に表示する

特定のウィンドウを、画面上で他のウィンドウより手前に表示させる方法について説明します。

Title

指定したフォームのタイトルを設定したり取得したりする方法について説明します。

Windows フォームでのデータ バインディング

Windows フォームでのデータ連結は、データ ソースの情報をフォーム上のコントロールで表示したり変更したりする手段を提供します。この点について具体的に説明します。従来のデータ ソースに対してだけでなく、データを含むほとんどすべての構造に対して連結できます。

System.Drawing.Imaging

イメージ描画のための高度なツールを備えた **System.Drawing.Imaging** について説明します。

方法 : アプリケーション アイコンを指定する

アイコンをアプリケーションにバインドする方法について説明します。

ローカリゼーション

方法 : AutoSize と TableLayoutPanel コントロールを使用して Windows フォームのローカリゼーションをサポートする

文字列の長さに応じた自動レイアウトを有効にする方法について説明します。

配置とローカリゼーション

ローカリゼーションのプランニングについて説明します。

メニューとコンテキスト メニュー

How to: Create Context Menus

ContextMenu コンポーネントを作成し、ユーザーが、使用頻度の高いメニュー コマンドを簡単に実行できるようにする方法について説明します。

方法 : Windows フォーム ContextMenu コンポーネントのメニュー項目を追加および削除する

Windows フォームのショートカット メニューの項目を追加する方法および削除する方法を説明します。

チュートリアル : ブックマークのショートカット メニューの作成

Bookmark コントロールのショートカット メニューを作成する方法について説明します。

印刷

方法 : PrintDialog コンポーネントを表示する

PrintDialog コンポーネントを表示する方法について説明します。

参照

関連項目

Visual J# での操作方法

XML Web サービス (Visual J# での操作方法)

このページでは、よく使用する J# XML Web サービス タスクに関するヘルプへのリンクを紹介します。その他、ヘルプでカバーされている一般的なタスク カテゴリについては、「Visual J# での操作方法」を参照してください。

チュートリアル : 分散アプリケーションの作成

多階層の分散アプリケーションを作成する方法について説明します。アプリケーションは、データ層、ビジネス オブジェクト層、およびユーザー インターフェイス層の 3 層で構成されます。

Acronym-WebService サンプル (XML Web サービスの作成と利用)

Web サービスの作成方法および利用方法について説明します。

ASP.NET を使用して作成した XML Web サービスのセキュリティ

ASP.NET がさまざまな認証オプションおよび承認オプションを提供するためにインターネット インフォメーション サービス (IIS: Internet Information Services) とどのように連動するかを説明します。

作成

XML Web サービスのチュートリアル (Visual J#)

Visual J# を使って単純な XML Web サービスを作成する手順について説明します。

アクセス

チュートリアル : Visual J# Web フォーム クライアントによる XML Web サービスへのアクセス

XML Web サービスの作成方法およびアクセス方法について説明します。

検索

Web サービス検出ツール (Disco.exe)

DISCO ツールを使用して Web サービスを検索します。

Visual J# の IDE の使用

Visual J# は、趣味や学術的な興味から、動的な Windows アプリケーションや Web サイトを構築したいと考えるユーザーのために、必要な機能だけを凝縮し軽快な動作を実現したツールです。

このセクションの内容

[新しいプロジェクト] ダイアログ ボックス (Visual J#)

プロジェクトの設定手順について説明します。

Visual J# のプロジェクト設定

プロジェクトの設定を変更するための手順について説明します。

方法 : Visual J# のナビゲーションと検索を実行する

オブジェクトやメソッドの検索手順について説明します。

方法 : コード スニペットを挿入する (Visual J#)

定義済みのコード ブロックを挿入することによりコーディングを効率化するための手順について説明します。

方法 : Visual J# でコードを入力する

アプリケーションに J# コードを追加する手順について説明します。

方法 : Visual J# コンソール アプリケーションを作成する

コンソール アプリケーションの作成手順について説明します。

方法 : Visual J# Windows アプリケーションを作成する

Windows フォームを使用して、Web ブラウザ アプリケーションを構築する手順について説明します。

関連するセクション

Visual J# コンパイラ

Visual J# リファレンス

[新しいプロジェクト] ダイアログ ボックス (Visual J#)

[新しいプロジェクト] ダイアログ ボックスには、あらかじめ定義された一連の Visual Studio プロジェクト テンプレートが表示されます。これらは、Visual Studio を使って作成できるプロジェクトの種類を表しています。プロジェクトの種類を選択すると、Visual Studio により、必要なコンパイラ オプションが設定され、基本的なアセンブリ参照がプロジェクトに追加されます。また、アイコン、カーソルなど、基になるすべてのコードおよびリソース ファイルが生成されるため、そのまま空のスタブ プロジェクトとしてコンパイルすることもできます。通常は、生成されたコードに独自の実装を追加した上で、そのプロジェクトをコンパイルします。

Visual J# には、プロジェクトの種類があらかじめ 4 つ定義されています。

プロジェクトの種類	**プロジェクトの説明**
Windows アプリケーション	Windows クライアント アプリケーションを作成する場合は、このプロジェクトの種類です。このプロジェクトを選択すると Windows フォームが作成されます。Windows フォームには、他のコントロールをドロップし、テキストやグラフィックスを表示できます。詳細については、「方法 : Visual J# Windows アプリケーションを作成する」を参照してください。
クラス ライブラリ	クラス ライブラリ プロジェクトでは、他のアプリケーションから参照可能なライブラリ ファイルが作成されます。
コンソール アプリケーション	コマンド ラインのユーティリティやアプリケーションを作成する場合は、このプロジェクトの種類です。テキスト ベースのターミナル ウィンドウで、プログラムが入出力されます。詳細については、「方法 : Visual J# コンソール アプリケーションを作成する」を参照してください。

参照

その他の技術情報

Visual J# の IDE の使用

Visual J# のプロジェクト設定

Visual J# の設定は、操作性を最適化し、J# を使った開発の生産性を最大化するように設計されています。Visual Studio の統合開発環境 (IDE) 設定は、必要に応じてカスタマイズしたり保存したりできます。

また、他のコンピュータの IDE 設定を移植または再読み込みすることもできます。詳細については、「方法 : コンピュータ間で設定を共有する」を参照してください。

ウィンドウおよびビュー

機能	既定で表示されるかどうか	備考
クラス ビュー	○	• フィルタ処理が有効です。 • クラス ビュー フォルダが有効です。 クラスの [実装されたインターフェイス] ノードを非表示にします。
コマンド ウィンドウ	×	ユーザー オプションとして表示されます。
[出力] ウィンドウ	○ (ビルド開始時に表示)	ショートカット キーを使用して GotoNextError コマンドを公開します。
ソリューション エクスプローラ	○	既定で、コード エディタとテキスト エディタの [選択したファイルと開いたファイルとを同期する] をオフにします。
スタート ページ	○	IDE を初めて起動した場合のみ表示されます。
タスク一覧 (Visual Studio)	×	ビルド コマンドが原因で表示されません。
ツールボックス	○	コンポーネントはアルファベット順で表示されます。

また、次の要素には、Visual J# プロファイルを選択したときに有効になる特定の動作があります。

ダイアログ ボックス

機能	動作
[クイック検索] ([検索と置換] ウィンドウ)	特定の文字列を 1 つまたは複数置き換えます。
[新しいプロジェクト] ダイアログ ボックス	コンピュータに他のプラットフォームをインストールすると、IDE により、対象のプラットフォームを指定できるコンボ ボックスが有効になります。
[オプション] ダイアログ ボックス (Visual Studio)	[その他のオプション] をクリックすると、ソース コントロール、デバイス ツール、データベース ツール、HTML デザイナ、および XML デザイナのオプションが表示されます。

キーボード

機能	動作
ショートカット キー	• 標準 VS エディタ • BRIEF エミュレーション • EMACS エミュレーション

開発環境の各種要素

機能	動作
Windows フォーム デザイナの設定	既定のビューは、コンポーネント デザイナ ビューではなく、コード エディタです。

メイン メニュー、ショートカット メニュー、ツール バー、コマンド	• 対応するショートカット キー ストロークがメイン メニュー項目の横に表示されます。
シェルの外観	Visual Studio に共通する外観と操作性を持ちます。
ウィンドウの管理とレイアウト	ウィンドウの既定のレイアウトは次のとおりです。 • ソリューション エクスプローラ、クラス ビュー、およびウィンドウは右側に、最大の長さで表示されます。 デバッガ ウィンドウの既定のレイアウトは次のとおりです。 • 既定では、[自動変数]、[ローカル]、および [ウォッチ 1] は IDE の一番下に表示されます。 • 呼び出し履歴およびイミディエイト ウィンドウは、独立したウィンドウで表示されるか、IDE の一番下に表示されます。 • 出力ウィンドウは表示されません。

参照

処理手順

方法 : チームの設定を指定する

関連項目

[ビルドの詳細設定] ダイアログ ボックス (J#)

[書式設定] ([オプション] ダイアログ ボックス - [テキスト エディタ] - [C#]/[J#])

方法：Visual J# コンソール アプリケーションを作成する

このトピックを読むと、Visual J# 開発環境を理解して、最も単純な Visual J# プログラムであるコンソール アプリケーションを構築できるようになります。コンソール アプリケーションでは、コマンド ラインですべての入出力が実行されます。そのため、コマンド ライン ユーティリティを記述する場合以外にも、言語の機能を簡単に試してみたい場合や、簡単な練習を行いたい場合などにも最適です。

メモ：
ここで説明する開発環境の特徴は、Windows フォーム アプリケーションを開発するときにも当てはまります。そのため、コンソール アプリケーションを作成する予定がない場合でも、このセクションを読み飛ばさないでください。

ここでは、次の項目について説明します。

- 新しいコンソール アプリケーションを作成するには
- ソリューション エクスプローラを表示するには
- コードの書式を維持するには
- IntelliSense を使用してコードの入力速度と精度を向上させるには
- アプリケーションをビルドして実行するには

このタスクでは、System.out Field を利用して、ディレクトリに含まれるすべてのファイル一覧とそのサイズを取得し、表示するプログラムを作成します。このコードを参考にして、ディレクトリ内で特定のファイル名を検索するユーティリティを作成することもできます。

次の例では、特定のディレクトリに格納されたファイルについての情報を表示するだけの、List Files という簡単なコンソール アプリケーションを作成します。

コンソール アプリケーションを作成するには

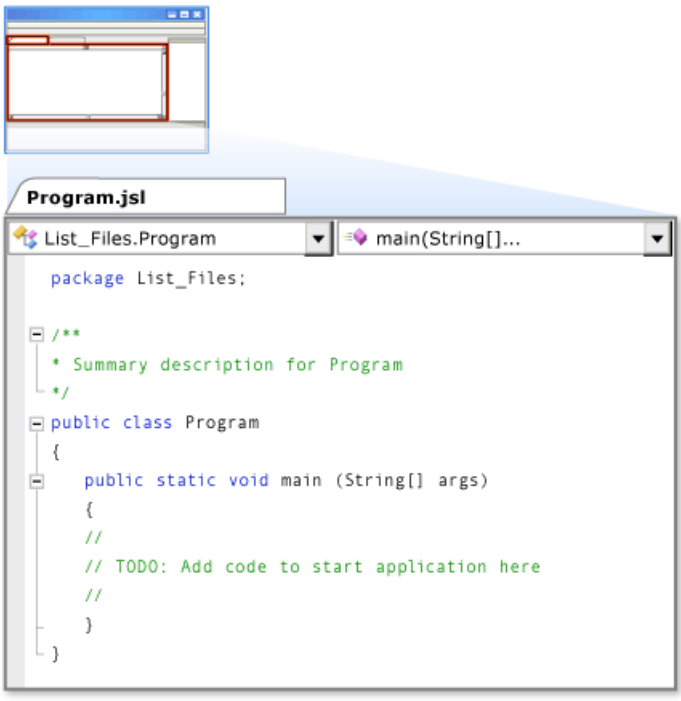
1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。

   [新しいプロジェクト] ダイアログ ボックスが表示されます。このダイアログ ボックスには、Visual J# で作成できる各種のプロジェクトが一覧表示されます。

2. [コンソール アプリケーション] テンプレートを選択し、アプリケーションの名前を「**List Files**」に変更します。

3. [OK] をクリックします。

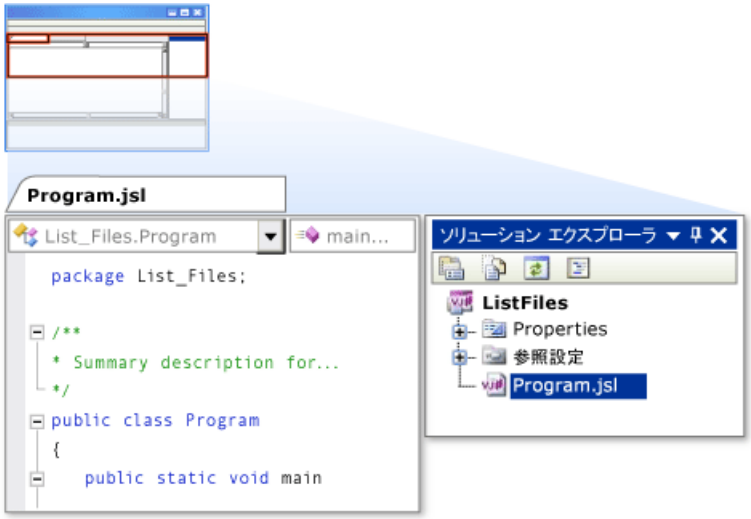
   指定したタイトルと同じ名前のプロジェクトが Visual J# によって作成され、コード ペインが表示されます。ここで、アプリケーションに必要な J# ソース コードを入力したり編集したりできます。プロジェクトは、実際に保存されるまで、一時的な場所に格納されます。



ウィンドウの上部にツール バーがあります。このツール バーには、プロジェクトの作成、読み込み、保存を行うアイコン、ソース コードを編集するアイコン、アプリケーションをビルドするアイコン、および Visual J# 環境を構築する他のウィンドウの表示/非表示を切り替えるアイコンが表示されます。ツール バーの右端にある 4 つのアイコンは、ソリューション エクスプローラやツールボックスなど、重要なウィンドウを開くときに使用します。いずれかのアイコンをポイントすると、対応するツールヒント ヘルプがポップアップ表示されます。

4. 画面右側の [ソリューション エクスプローラ] タブまたはツール バーの [ソリューション エクスプローラ] アイコンをクリックします。

   ソリューション エクスプローラは、プロジェクトを構成する多様なファイルが表示される便利なペインです。このプロジェクトで最も重要なファイルは、アプリケーションのソース コードが記述された Program.jsl です。

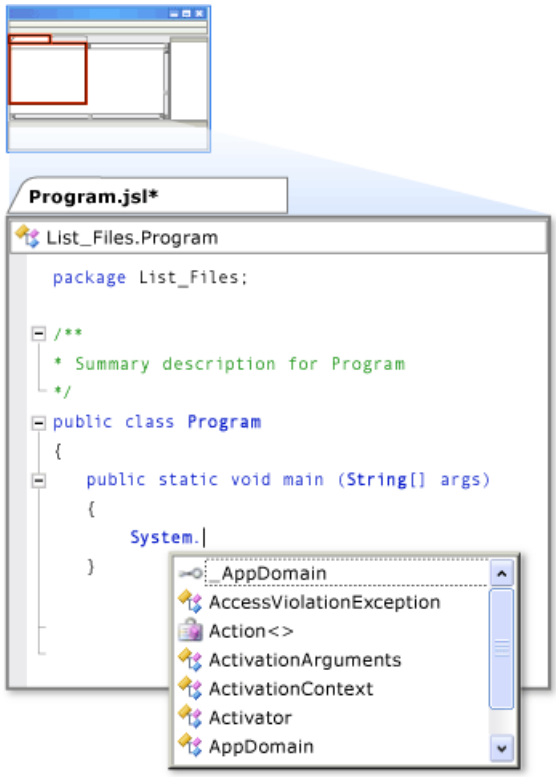


5. main メソッド内にある左の中かっこ ({) の右をクリックし、Enter キーを押して新しい行を追加します。

   カーソル位置が自動的にインデントされます。

6. クラス名「**System.**」をコード エディタに入力します。

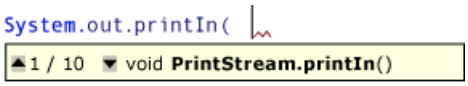
   J# クラスの名前やキーワードを入力するには、2 つの方法があります。単語全体を自分で入力する方法と、コード エディタの一機能である IntelliSense ツールで自動入力する方法です。たとえば、「**sys**」と入力すると、IntelliSense で予測された入力候補がポップアップ リストに表示されます。これだけでは "System" という単語が表示されないため、リストをスクロール ダウンするか、"system" の残りの文字の入力を続けます。"system" がリストで強調表示された状態で Enter キーまたは Tab キーを押すか、ダブルクリックすると、`System` がコードに追加されます。



   IntelliSense を使用するメリットは、大文字と小文字の区別とスペルを正しく入力できる点です。コードを自分で入力するか、IntelliSense で自動入力するかは、自由に選択できます。

7. ピリオドおよびクラス名 **out** を入力した後、ピリオドをもう 1 つ入力します。

   out の後にピリオドを入力すると、別の IntelliSense リストが表示されます。このリストには、**out** ストリーム クラスに含まれるクラス、メソッド、およびプロパティがすべて表示されます。PrintStream.println Method メソッドは、リストの一番下にあります。「**println**」の入力を完了するか、表示されるまで↓キーを押して、Enter キー、Tab キー、またはダブルクリックのいずれかを操作します。**println** がコードに追加されます。

8. 左かっこ (() を入力します。他の IntelliSense 機能であるメソッドのシグネチャが、ツールヒント メッセージに表示されます。この例では、10 種類のシグネチャが表示されます。↑キーと↓キーをクリックしてリストを確認できます。

9. 文字列 **"This program lists all the files in the directory"** を入力します。

   引用符内にメッセージを入力し、閉じかっこ ()) を追加します。何かが欠落していることを示す赤色の波線が表示されます。セミコロン (;) を入力すると、この波線は表示されなくなります。

10. プログラムを完成させます。

    次のコードを入力するか、コピーして貼り付けて、プログラムを完成させます。

```
import java.io.File;

public class Program
{
    public static void main(String[] args)
    {
        System.out.println("This program lists all the files in the directory:");

        File filePath = new File("c:\\Program Files");
        java.io.File[] fileList = filePath.listFiles();
        for (int idx = 0; idx < fileList.length; idx++)
        {
            System.out.println(fileList[idx]);
        }
    }
}
```

11. プログラムを実行します。

    ここまでの操作で、初めてのプログラムが完成し、コンパイルと実行の準備が整いました。コンパイルして実行するには、F5 キーを押すか、ツール バーの [開始] アイコンをクリックします。

    プログラムをコンパイルして実行すると、[コンソール] ウィンドウが開き、ファイルとそのサイズの一覧が表示されます。Enter キーを押すと、プログラムが終了します。

12. J# プログラムの初心者の方は、この時点で「Visual J# リファレンス」を参照し、いくつかのサンプル コードを試すことをお勧めします。Visual J# 開発環境の詳細と、Windows アプリケーションの作成方法については、次の「方法 : Visual J# Windows アプリケーションを作成する」を参照してください。

参照

その他の技術情報

Visual J# の IDE の使用

方法：Visual J# でコードを入力する

コード エディタを使用して、Visual J# アプリケーションにコードを挿入する方法を次の例に示します。新しい .jsl コード ファイルを作成した場合と、既存のファイルを開いて使用する場合の 2 とおりの方法を紹介します。

新しいプロジェクトを作成するには

1. [ファイル] メニューの [新規作成] をポイントし、[新しいプロジェクト] をクリックします。

   [新しいプロジェクト] ダイアログ ボックスが表示されます。このダイアログ ボックスには、Visual J# で作成できる各種のプロジェクトが一覧表示されます。

2. [コンソール アプリケーション] テンプレートを選択し、アプリケーションの名前を「**CreateCode**」に変更して [OK] をクリックします。

   指定したタイトルと同じ名前のプロジェクトが Visual J# によって作成され、コード ペインが表示されます。ここで、アプリケーションに必要な J# ソース コードを入力したり編集したりできます。プロジェクトは、実際に保存されるまで、一時的な場所に格納されます。

   Program.jsl 内であらかじめ生成されたコードは、わかりやすいように色付きで表示されます。既定では、青色で表示されるテキストとして、アプリケーションのキーワード、アクセス修飾子、メソッドの戻り値の型などがあります。たとえば、`public class` というフレーズでは、だれでも使用できるクラスとして `Program` を定義しています。

   データ型、メソッド、関数、ネイティブ データ型などは黒色の太字で表されます。たとえば、J# のコンソール アプリケーションの場合、Program が既定のクラス名になってしまうこともあります。この名前は、適宜、自分のアプリケーションにふさわしい名前に変更できます。

   緑色のテキストは、その情報に、有用なコメントや javadoc ドキュメント コメントが含まれていることを示します。

3. Program.jsl で、`// TODO: Add code to start application here` 直下のコメント付きの行をクリックし、Enter キーを押します。

4. 空白行をクリックし、コードの入力を開始します。

既存の J# ファイルにコードを入力するには

1. ソリューション エクスプローラで、.jsl ファイルをダブルクリックします。

   または

   [ファイル] メニューの [開く] をポイントし、[ファイル] をクリックします。ファイルを検索する対象フォルダをクリックします。

   または

   Ctrl キーを押しながら O キーを押し、開くファイルをダブルクリックします。

   目的のファイルが コード ペインに表示されます。

2. 空白行をクリックし、コーディングを開始します。

   または

3. 任意の行の先頭でクリックし、Enter キーを押します。挿入された空白行からコーディングを開始します。

参照

その他の技術情報

Visual J# の IDE の使用

方法：Visual J# Windows アプリケーションを作成する

このトピックを読むと、Visual J# 開発環境の要素を理解して、Windows フォームを使用した簡単な J# プログラムを構築できるようになります。Windows フォームを使用すると、ダイアログ ボックス、メニュー、ボタンなどのコントロールをはじめ、標準的な Windows アプリケーションのユーザー インターフェイスを構成するコンポーネントが含まれたプロジェクトを作成できます。

この例では、よくアクセスする Web サイトがあらかじめ登録された、独自の Web ブラウザ アプリケーションを作成する方法について説明します。お気に入りの Web サイトを登録することによって、自由にカスタマイズできます。

メモ：
ここで説明する開発環境の特徴は、その多くがコンソール アプリケーションを開発するときにも当てはまります。そのため、Windows アプリケーションを作成する予定がない場合でも、このセクションを読み飛ばさないでください。

ここでは、次の項目について説明します。

- 新しい Windows アプリケーションを作成するには
- コード ビューとデザイン ビューを切り替えるには
- Windows フォームのプロパティを変更するには
- メニューのコントロールを追加するには
- [ComboBox] コントロールを作成して設定するには
- WebBrowser コントロールを使用するには
- コントロールのハンドラを作成するには

J# Windows アプリケーションを作成するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。

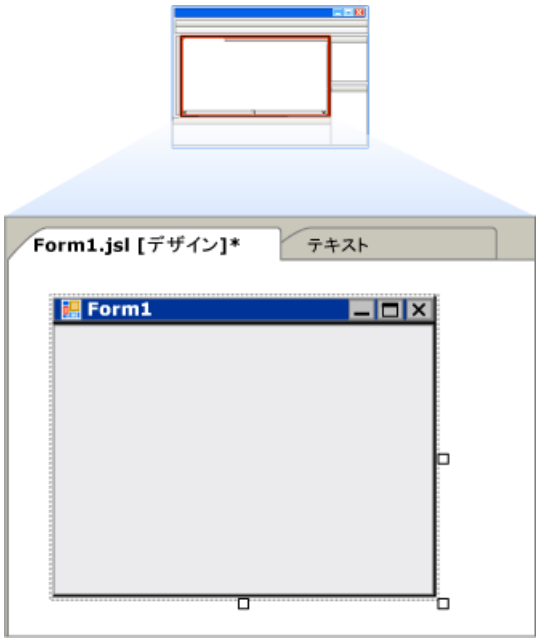
   [新しいプロジェクト] ダイアログ ボックスが表示されます。このダイアログ ボックスには、Visual J# で作成できる既定のプロジェクトの種類が一覧表示されます。

2. プロジェクトの種類として [Windows アプリケーション] を選択します。

3. [プロジェクト名] ボックスに「**Web Browser**」と入力して、アプリケーションの名前を変更します。

4. [OK] をクリックします。

   指定したタイトルと同じ名前のプロジェクトが Visual J# によって作成され、Visual J# の Windows フォーム デザイナが開きます。既定のビューは、新しい Windows フォーム (Form1) になります。F7 キーを押すか、[表示] メニューの [コード] を選択することにより、このビューをソース コード ビューに変更することもできます。Shift キーを押しながら F7 キーを押すと、[デザイナ] ビューに戻ることができます。



Form1 は、アプリケーションが起動されたときに表示されるフォームです。このフォームには、さまざまなコントロールを追加できます。ユーザーからの入力を受け付けたり、テキストやグラフィックを表示したりできるほか、Windows アプリケーションで実行できるすべてのことが可能となります。また、ウィンドウを管理するために必要なコードが、Visual J# によって自動的に作成されます。J# では、部分型はサポートされないため、すべてのコードがフォームのメイン ファイルに格納されます。

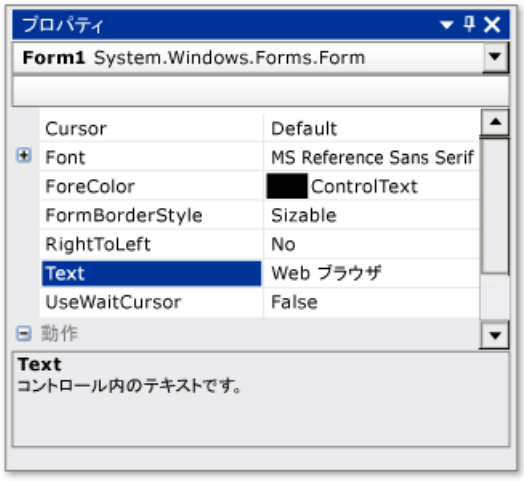
5. Windows フォームのサイズを変更します。

   [デザイナ] ビューが表示されていない場合は、Shift キーを押しながら F7 キーを押してください。Windows フォームの右下隅をクリックし、カーソルの形が 2 方向矢印に変わったら、フォームの隅をドラッグし、画面の 4 分の 1 になるように幅と高さを調整します。このウィンドウには、Web サイトが表示されます。これは、画面を狭くしないための操作です。

6. Windows フォームのタイトルを変更します。

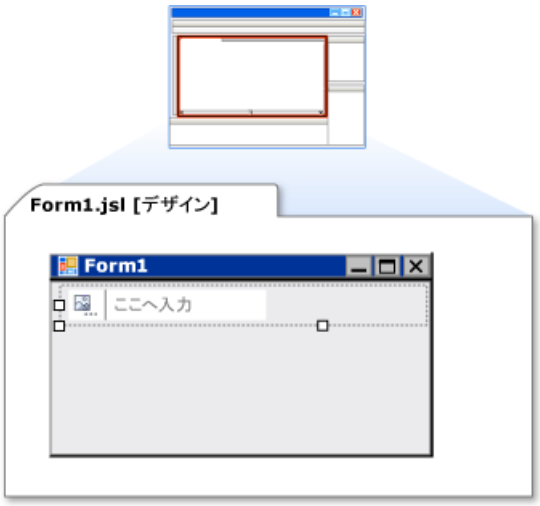
   [プロパティ] ウィンドウが表示されていない場合は、[表示] メニューの [プロパティ ウィンドウ] を選択します。このウィンドウには、現在選択されている Windows フォームまたはコントロールのプロパティが表示されます。また、既存の値を変更することもできます。

   Text というプロパティを見つけます。右側の列で Form1 を選択し、「**Web Browser**」と入力します。Enter キーまたは Tab キーを押して、コントロールからフォーカスを移動すると、Windows フォーム上の名前が変更されていることがわかります。[(オブジェクト名)] という同様のフィールドがあることに注目してください。これは、Windows フォームを格納しているクラスの名前です。この名前も変更できます。ただし、以降の説明では、Form1 をそのまま使用しています。クラス名を変更した場合は、適宜、クラス名の部分を置き換えて参照してください。



7. メニュー コントロールを追加します。

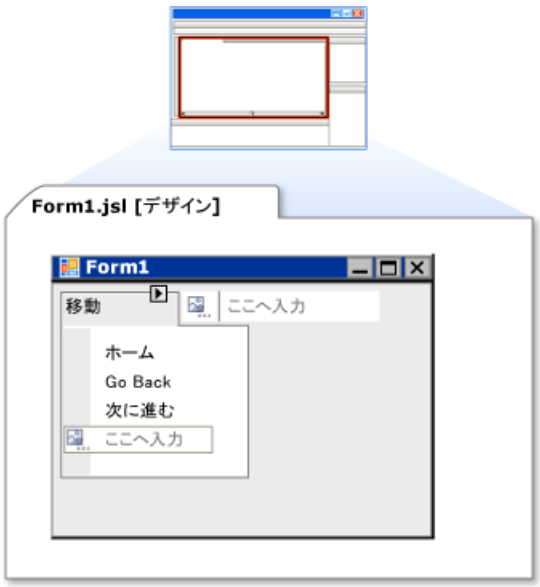
   [表示] メニューの [ツールボックス] を選択して、[ツールボックス] を開きます。MenuStrip が表示されるまで、コントロール一覧を下にスクロールします。このコントロールを Windows フォーム上の任意の場所にドラッグします。



フォーム上に MenuStrip コントロールをドロップすると、フォーム上部に既定のメニューが作成されます。

8. [メニュー] の内容を設定します。

   このボックスの "ここへ入力" と表示されている部分に、「**Navigate**」と入力してメニュー名を設定します。Enter キーを押すと、新しい空のボックスが表示され、他のメニューおよびメニュー項目を作成できる状態になります。下の方のボックスに、「**Home**」と入力します。Enter キーを押すと、別のボックスが表示されます。「**Go Back**」と入力します。Enter キーを押し、「**Go Forward**」と入力します。以上のメニュー項目で、基本的な Web サイト ナビゲーション コントロールが作成されました。



   ソース コードの可読性を高めるには、メニュー項目に割り当てられている既定の名前を変更する必要があります。[プロパティ] ウィンドウを開いて各メニュー項目を順番にクリックすると、menuItem1 や menuItem2 などのデザイン名が表示されることがわかります。

   メニュー項目の名前を変更し、[Home] には menuItem_Home を、[Go Back] には menuItem_GoBack を、[Go Forward] には menuItem_GoForward という名前をそれぞれ指定します。

9. ボタンを追加します。

   [ツールボックス] にあるボタン コントロールを、Windows フォームでメニュー バー直下のほぼ中間にドラッグします。このボタンを [プロパティ] ウィンドウで表示し、Text プロパティを button1 から Go に変更します。また、デザイン名を button1 から button_go に変更します。

10. ComboBox を追加します。

    [ツールボックス] の ComboBox コントロールをドラッグし、新しいボタンの左にドロップします。次のスクリーンショットに示したように、ComboBox の端と隅をドラッグし、ボタンと揃うようにサイズと位置を変更します。

メモ：
Windows フォームでコントロールを移動すると、青い線が表示されます。この線は、コントロールの縦横を整列するときに補助するガイドです。複数のコントロールを選択して整列することもできます。この場合、フォーム内をドラッグし、コントロールを選択ボックスで囲むか、Shift キーを押しながらコントロールをクリックします。[レイアウト] ツール バーに用意された、各種の整列ボタンとサイズ変更ボタンを使用して、コントロールのレイアウトを調整します。

11. ComboBox を設定します。

    ComboBox には、選択肢のドロップダウン リスト機能があります。このプログラムでは、ComboBox にお気に入りの Web サイト一覧を表示して、簡単にアクセスできるようにします。

    サイト一覧を作成するには、ComboBox をクリックしてプロパティを表示します。Items プロパティの横の列をクリックすると、省略記号のボタン (...) が表示されます。このボタンをクリックして、[文字列コレクション エディタ1] ダイアログ ボックスを開きます。このダイアログ ボックスで、ComboBox に表示される内容を追加できます。Web サイトの URL を入力し、Enter キーを押します。URL はいくつでも追加できます。

12. Web ブラウザ コントロールを追加します。

    [ツールボックス] を開き、スクロール ダウンして Web ブラウザ コントロールを表示します。コントロールを Windows フォームにドラッグします。Windows フォーム内で、WebBrowser コントロールのサイズを適宜変更します。このとき、ComboBox コントロールと Button コントロールとが重ならないようにします。WebBrowser コントロールのサイズを変更しづらいときは、プロパティを開いて、[Dock] が [None] に設定されていることを確認します。[Anchor] を [Top, Bottom, Left, Right] に設定した場合、アプリケーションのウィンドウ サイズを変更すると、WebBrowser コントロールも適切なサイズに調整されます。

    Web ブラウザ コントロールを使用すると、Web ページのレンダリングに伴う困難な作業のすべてを実行できます。

13. Button コントロールのハンドラを追加します。

    ここまでの操作で、アプリケーションのデザイン段階が完了しました。これから、J# コードを追加して、プログラムの機能を組み込む段階に入ります。

    まず、ボタンと各メニュー オプションにハンドラを追加する必要があります。ハンドラは、特定のコントロールがアクティブになったときに実行されるメソッドです。Visual J# では、空のハンドラが自動的に作成されます。

    ボタンをダブルクリックすると、プロジェクトのコード エディタが表示されます。また、クリック イベント (ユーザーがボタンをクリックしたときに生成されるイベント メッセージ) のハンドラが自動的に生成されていることがわかります。次のコードを参考にして、ハンドラ メソッドにコードを追加します。

```
private void button_go_Click(Object sender, System.EventArgs e)
    {
      webBrowser1.Navigate(comboBox1.get_SelectedItem().toString());
    }
```

    このコードでは、ComboBox コントロールで現在選択されている項目 (Web URL を含む文字列) を取得し、Web ブラウザのナビゲーション メソッドに渡します。ナビゲーション メソッドにより、該当する URL の Web ページの内容が読み込まれ、表示されます。

14. メニュー オプションのハンドラを追加します。

    F7 キーを押して [デザイナ] ビューに戻り、各メニュー項目を順にダブルクリックしてください。Visual J# によって、各メニュー項目のハンドラ メソッドが作成されます。次のコードを参考にして、メソッドを編集します。

```
private void menuItem_Home_Click(Object sender, System.EventArgs e)
    {
        // Go to Home Page menu option.
        webBrowser1.GoHome();
    }
private void menuItem_GoForward_Click(Object sender, System.EventArgs e)
    {
        // Go forward menu option.
        webBrowser1.GoForward();
    }
private void menuItem_GoBack_Click(Object sender, System.EventArgs e)
    {
        // Go back menu option.
        webBrowser1.GoBack();
    }
```

    各メニューのイベント ハンドラから、Web ブラウザ クラスのナビゲーション メソッドが呼び出されます。

メモ：

このコードを見ると、メニュー オプションに付けられた既定の名前が紛らわしくなっていることがわかります。そのため、各メニュー コントロールを作成するときに、[プロパティ] ウィンドウで名前を変更することをお勧めします。ハンドラの名前に、メニュー オプションの名前が反映されます。

15. 初期化コードを追加します。

    最後に、`Form1` メソッドに、必要なコードを追加します。このメソッドは、`Form1` オブジェクトのコンストラクタです。Windows フォームが作成されたときに呼び出されます。したがって、コントロールの初期値または他の設定を変更する必要がある場合、必要なコードをこの位置に追加することになります。

    ここでは、Web Browser コントロールに対し、コンピュータの既定のホーム ページが表示されるように設定し、さらに、ComboBox の初期値を設定することにします。次のコードを Form1 メソッドに追加してください。

```
public Form1()
    {
        InitializeComponent();
        comboBox1.set_SelectedIndex(0);
        webBrowser1.GoHome();
    }
}
```

16. プログラムをビルドして実行します。

    F5 キーを押してビルドし、Web ブラウザを実行します。画面に Windows フォームが表示され、次にコンピュータの既定のホーム ページが表示されます。ComboBox コントロールを使用して Web サイトを選択し、[Go] をクリックすると、そのサイトが表示されます。メニュー オプションでホーム ページに戻ったり、以前に表示した Web サイトと現在の Web サイト間で移動することもできます。

17. Visual J# プログラミングの初心者の方は、この時点で「Visual J# リファレンス」を読むことをお勧めします。Visual J# 開発環境の詳細と、特に IntelliSense を使用したコンソール アプリケーションの作成方法については、前のセクションの「方法 : Visual J# コンソール アプリケーションを作成する」を参照してください。

参照

その他の技術情報

Visual J# の IDE の使用

方法：Visual J# のナビゲーションと検索を実行する

コード エディタでは、マウスや移動キーを使ったさまざまな方法でテキスト間またはコード間を移動できます。

- 方向キーを使用して一度に 1 文字ずつ移動するか、**Ctrl** キーを押しながら方向キーを押して一度に 1 単語ずつ移動します。方向キーを押して一度に 1 行ずつ移動することもできます。
- 目的の位置をマウスでクリックします。
- スクロール バーを使用するか、マウスのスクロール ボールを使用して、テキスト間を移動します。
- **Home**、**End**、**PageUp**、**PageDown** の各キーを使用します。
- **Ctrl** キーを押しながら **PageUp** キーまたは **PageDown** キーを押し、それぞれウィンドウの最上部または最下部にカーソルを移動します。
- **Ctrl** キーを押しながら↑キーまたは↓キーを押し、カーソルを移動せずにビューをスクロールします。
- アクティブ ドキュメントにおいて、挿入ポインタを直前の位置に移動するには [戻る] ボタンを、最近使用した位置に戻るには [次に進む] ボタンを使用します。

[インクリメンタル検索]、[指定行へのジャンプ]、[定義へ移動]、[ドキュメントの先頭]、[文書の最後]、[貼り付け]、[ファイルの挿入] を使用すると、挿入ポインタをアクティブ ドキュメント内の離れた場所に移動できます。[戻る] ボタンおよび [次に進む] ボタンでは、挿入ポインタの位置を 20 個まで保持できます。

次のセクションでは、特定のセクションおよびコード行に移動するためのその他の方法について説明します。

- [ナビゲーション バー]
- [ブックマーク] ウィンドウのブックマーク
- タスク一覧内のコメント タスク
- インクリメンタル検索
- [指定行へのジャンプ] コマンド
- [定義へ移動] コマンド

ナビゲーション バーによる移動

ナビゲーション バーには 2 つのコンボ ボックスがあり、どちらもコード エディタの上部に表示されます。ナビゲーション バーを使用すると、特定のクラス/型またはその中のプロシージャ/メンバに直接移動できます。すべてのプロジェクトにナビゲーション バーがあるわけではありません。コード内を移動する方法について、次に説明します。

メモ：
ナビゲーション バーの 2 つのコンボ ボックスの名前は、プロジェクトの種類によって異なります。たとえば、Visual Basic プロジェクトのボックスの名前は、[クラス名] と [メソッド名] です。C# プロジェクトのボックスの名前は、[型] と [メンバ] です。

コード エディタからナビゲーション バーにフォーカスを移動するには

- Ctrl キーを押しながら F2 キーを押します (ショートカット キー)。

ナビゲーション バーからコード エディタにフォーカスを戻すには

- Esc キーを押します。

ナビゲーション バーの 2 つのコンボ ボックス間でフォーカスを切り替えるには

- **Tab** キーを押します。

フォーカスのあるナビゲーション バーの項目を選択し IDE に戻るには

- **Enter** キーを押します。

クラスまたは型に移動するには

- ドキュメントの左上隅にある [クラス名] ボックスまたは [型] ボックスの一覧で、名前をクリックします。

クラスのプロシージャに直接移動するには

- ドキュメントの右上隅にある [メソッド名] ボックスまたは [メンバ] ボックスの一覧で、プロシージャをクリックします。

ブックマークを使用した移動

ブックマークを設定すると、コードのセクションへ簡単に移動できます。Visual J# Express Edition におけるブックマークの使用方法を次の例に示します。

ブックマークを追加するには

1. エディタ内で、後で戻る行を選択します。
2. [ブックマーク] ツール バーの [現在行にブックマークを追加/削除します。] をクリックします。

   または

   Ctrl キーを押しながら K キーを押します。

   ブックマーク記号がコード エディタの左端の余白に表示されます。

   
3. ブックマークのオンとオフを切り替えるには、Ctrl + K キーを押します。

アクティブ ドキュメントに設定されているブックマーク間を移動するには

1. コード内の次のブックマークまでスクロールするには、[ブックマーク] ツール バーの [カレットを次のブックマークへ移動します。] をクリックします。

   または

   Ctrl キーを押しながら K キーを押し、次に Ctrl キーを押しながら N キーを押すと、次のブックマークに移動します。

2. コード内の前のブックマークまでスクロールするには、[ブックマーク] ツール バーの [カレットを前のブックマークへ移動します。] をクリックします。

   または

   Ctrl キーを押しながら K キーを押し、次に Ctrl キーを押しながら P キーを押すと、前のブックマークに移動します。

   元の場所に戻るには、[戻る] や [次に進む] を使用する方法もあります。

ドキュメントからすべてのブックマークを削除するには

- [ブックマーク] ウィンドウで [ブックマークをクリア] をクリックします。

  または

  Ctrl キーを押しながら K キーを押し、次に Ctrl キーを押しながら L キーを押すと、アクティブ ドキュメントからブックマークがすべて削除されます。

メモ：

単一のブックマークを削除するには、目的のブックマークを右クリックし、[ブックマークの設定/解除] をクリックするか、ブックマークが削除されるまで Ctrl + K キーを繰り返し押します。

コメント タスクを使用したナビゲーション

コメント タスクは `//TODO` というフレーズで始まり、完了することが必要なコードの位置がわかります。コメント タスクをコード中に作成し、戻りたい位置に印を付けておくことで、タスク一覧からこの位置に直接移動できます。

コメント タスクを作成して使用するには

1. [表示] メニューの [その他のウィンドウ] をポイントし、[タスク一覧] をクリックします。

   [タスク一覧] のグリッドには、作業項目とそのステータスの一覧が表示されます。

2. [コメント] ボックスの一覧の [コメント] を選択します。

   アクティブ ドキュメントのコードに存在するすべてのコメント タスクが表示されます。

3. コードに 2 つのスラッシュ (/) の後に 1 つのスペースとキーワード TODO を入力し、コメント タスクを追加します。たとえば、「// TODO このコードを完了させること」のように入力します。

   「TODO」と入力すると、このタスクは [タスク一覧] の [コメント] リストに自動的に追加されます。コメント タスクのテキストを修正すると、[コメント] リストは自動的に更新されます。

4. コメント タスクをクリックすると、コード内の該当位置にジャンプします。

   詳細については、「タスク一覧 (Visual Studio)」を参照してください。

インクリメンタル検索

インクリメンタル検索では、検索文字列を入力すると、現在のドキュメントにある検索文字列の場所に直接移動できます。インクリメンタル検索の実行手順を次に説明します。

インクリメンタル検索を使用して移動するには

1. [編集] メニューの [詳細] を選択し、[インクリメンタル検索] を選択します。
2. 検索対象の文字列の入力を開始します。

   文字を追加するごとに、現在のドキュメントで見つかった最初の一致項目がコード エディタで強調表示されます。

[指定行へのジャンプ] コマンド

[指定行へのジャンプ] コマンドを使用して、挿入ポインタを特定の行番号に移動させる手順を次に説明します。

特定の行番号にジャンプするには

1. [全般] ([オプション] ダイアログ ボックス - [テキスト エディタ] - [すべての言語])で、[行番号] オプションを選択します。
2. エディタでドキュメントを開いた状態で、[編集] メニューの [指定行へのジャンプ] を選択します。
3. 表示する行番号を入力します。

[定義へ移動] コマンド

[定義へ移動] コマンドを使用すると、ローカル クラス宣言やメソッドのプロシージャ、または変数定義にジャンプします。たとえば、`MyProc()` というプロシージャが定義されているとき、後で自分のコードから `MyProc()` を呼び出す場合は、[定義へ移動] を使用してこのメソッドのプロシージャに直接移動できます。

[定義へ移動] コマンドを使用して、ローカル クラスの宣言、メソッド、または変数の各定義を検索する手順を次に説明します。

呼び出し先のプロシージャまたは変数の定義に移動するには

1. プロシージャまたは変数の呼び出し箇所を選択します。
2. F12 キーを押します。

   コード エディタにプロシージャまたは変数が表示されます。

3. 呼び出し元に戻るには、[戻る] ボタンを押します。

   定義がプロジェクトの外部に存在する場合は、その呼び出し先がオブジェクト ブラウザに表示されます。詳細については、「Using the Object Browser」を参照してください。

参照

その他の技術情報

Visual J# の IDE の使用

方法：コード スニペットを挿入する (Visual J#)

IntelliSense コード スニペットの挿入は、コード エディタのナビゲーション ツールを使用するか、または Windows エクスプローラからファイルをドラッグすることにより行うことができます。挿入するスニペットの名前がわかっている場合には、ショートカットを入力することで挿入できます。目的に合ったスニペットを参照する必要がある場合には、スニペット ピッカーを使用できます。スニペット ピッカーでは、スニペットがカテゴリごとに一覧表示されており、その中から選択できます。

コード スニペットは、3 とおりの方法で挿入できます。Intelisense メニューを使用する方法、ショートカットを入力する方法、および、Windows エクスプローラからドラッグする方法です。IntelliSense コード スニペットを挿入するための手順を次に示します。

スニペットをコード エディタで参照して挿入するには

1. コードの挿入位置で [コード エディタ] を右クリックし、[Intelisense] をポイントします。
2. ショートカットの [拡張] をクリックします。

   または

   コード エディタで Tab キーに続けて疑問符を入力し、コード スニペット ピッカーを開きます。
3. スニペット ピッカーで目的のタスクに移動し、それをクリックします。

   コードにスニペット コードが挿入されます。コードに追加されたスニペットは、コード エディタを使ってカスタマイズできます。たとえば、変数名をより適切な名前に置き換えることができます。

ショートカットがわかっているスニペットを挿入するには

- [コード エディタ] で、スニペットのショートカットを入力した後に Tab キーを押します。スニペットが挿入されます。

  または
- [コード エディタ] で、スニペットのショートカットの先頭の数文字を入力した後、Tab キーを押します。ピッカーが表示され、入力した文字でショートカットが始まるスニペットの一覧が示されます。

Windows エクスプローラを使用してコードにスニペットを挿入するには

1. Windows エクスプローラを開きます。
2. Windows エクスプローラで、挿入するファイルが格納されているフォルダに移動します。Visual Studio 2005 を既定の位置にインストールした場合は、スニペットは C:\Program Files\Microsoft Visual Studio 8\VJ#\Expansions\ に格納されています。
3. Windows エクスプローラから、コード内の挿入位置にファイルをドラッグします。

参照

その他の技術情報

Visual J# の IDE の使用

Visual J++ 6.0 からのアップグレード

このセクションでは、Microsoft® Visual J++ 6.0® アプリケーションを Microsoft Visual J# にアップグレードするための推奨事項と手順について説明します。ここには、計画、トラブルシューティング、リソースの使用、Java 言語/COM 相互運用機能を使うプロジェクトのアップグレード、およびさまざまな com.ms.* パッケージからのアップグレードに関するトピックが含まれています。また、Visual J# アップグレード ウィザードとアップグレード レポートの使い方に関するトピックもあります。

既存の Visual J++ 6.0 プロジェクトは、Visual J# で開いてアップグレードできますが、アップグレード後にプロジェクトを変更することが必要な場合もあります。

このセクションの内容

Visual J++ ユーザーへの Visual J# の紹介

Visual J# によって .NET Framework に Java 言語のサポートが追加される方法について説明します。

アップグレードの計画

Visual J++ から Visual J# へのアップグレードの簡単な概要を示します。

Visual J# アップグレード ウィザード

Visual J# アップグレード ウィザードの概要を示し、その機能の使い方について詳しく説明します。

Visual J# のアップグレード レポート

アップグレード レポートの形式と目的について説明します。

Visual J++ 6.0 プロジェクトのアップグレード

アップグレード ウィザードを使って Visual J++ 6.0 プロジェクトを Visual J# にアップグレードする方法について説明します。

リソースを使う Visual J++ 6.0 アプリケーションのアップグレード

Visual J++ 6.0 プロジェクトで使用されるリソースを .NET Framework リソースに変換する手順について説明します。また、リソースの埋め込みやリンクに関連する互換性の問題についても示します。

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

Visual J++® 6.0 を使って開発された Java 言語/COM コンポーネントをコンパイル、デバッグ、および実行する方法について説明します。

クラスを動的に読み込むアプリケーションのアップグレード

CLASSPATH 変数とバインディングに関するセクションがあります。

カスタム クラス ローダーを使うアプリケーションのアップグレード

Visual J# でカスタム クラス ローダーを使う場合に指定できない引数を示します。

Visual J++ 6.0 のセキュリティのセマンティクスを使うアプリケーションのアップグレード

セキュリティのセマンティクスと、サードパーティのプロバイダを使うアプリケーションについて説明します。

com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード

よく使用される com.ms.* パッケージの一部と同等の .NET Framework クラス ライブラリを示します。com.ms.xml.*、com.ms.mtx、com.ms.wfc.html の各パッケージからのアップグレードに関するトピックが含まれています。

Visual J++ 6.0 プロジェクトからのアップグレードに関する一般的な問題

プロジェクトを Visual J# にアップグレードした後に発生する可能性のある問題の一覧です。

条件付きデバッグのサポート

Visual J++ 6.0 の **@conditional** 属性と .NET の ConditionalAttribute クラスを使用した場合の違いについて説明します。

Visual J++ 6.0 と Visual J# の同時利用

Visual J++ 6.0 ソリューションをアップグレードした後に Visual J++ 6.0 と Visual J# を並行して使う方法を説明します。

関連するセクション

言語サポート

Visual J++ 6.0 と .NET Framework の Visual J# でサポートされている機能とサポートされていない機能に関するリファレンスです。

クラス ライブラリのサポート

パッケージのサポートの有無、およびクラス ライブラリのサポートに関する実装固有の詳細情報の一覧です。

Visual J# のアップグレード リファレンス

Visual J++ 6.0 のプロジェクトを Visual J# にアップグレードするときにアップグレード ウィザードで検出される問題のヘルプが用意されています。

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

Java 言語/COM 相互運用機能を使う Visual J++ 6.0 のプロジェクトをアップグレードする場合の考慮点について説明します。

Visual J#

Visual J# ドキュメントのさまざまなトピックへのリンクの一覧です。

Visual J# の移行

Java と J++ のアプリケーション、アプレット、およびバイトコードを Visual J# に変換するためのツールについて説明します。

Visual J++ ユーザーへの Visual J# の紹介

Visual J# 製品の特徴を次に示します。

- .NET Framework に Java 言語のサポートを追加します。この製品は、Java 言語のプログラマに .NET Framework の充実した機能を提供し、既存の Java 言語の技術とコードをそのまま使用できるようにします。
- XML Web サービスのビジョンを理解し、ASP.NET、ADO.NET、Windows フォームの各ライブラリの豊富なプログラミング フレームワークを利用して魅力的なアプリケーションを作成できます。
- Visual J++ の開発者にシームレスなアップグレード方法を提供します。Visual J# は、Visual Studio と完全に統合されています。Visual J# には、Visual J++ の開発者が使い慣れたインターフェイスが用意されています。

製品の概要については、「Visual J# の概要」を参照してください。

Visual J# と Visual J++ は、1 つのシステムに共存させることができます。アップグレードを行っても、プロジェクトのソース ファイルとその他のファイルは変更されません。また、既存のプロジェクト ファイルも変更されません。Visual J# にアップグレードした後も、既存の Visual J++ プロジェクトを開き、Visual J++ でアプリケーションを操作できます。

Visual J# へのアップグレードに関する重要事項を次に示します。

- Visual J# での開発は、.NET Framework を対象としています。つまり、Visual J# プロジェクトの出力は、Microsoft Intermediate Language (MSIL) のバイナリです。これらのバイナリは、Visual J# の実装に用意されているランタイムのサポートを使って、共通言語ランタイム (CLR: Common Language Runtime) で実行されます。
- Visual J# の実装には、JDK 1.1.4 ライブラリの機能の大部分が含まれています。詳細については、「クラス ライブラリのサポート」を参照してください。
- Visual J# コンパイラは、言語の主要機能をすべてサポートしており、.NET Framework の使用も完全にサポートしています。また、J/Direct や Java 言語/COM 相互運用機能など、Visual J++ でサポートされている拡張機能もサポートしています。
- WFC プロジェクトでのデザイン時サポートはありません。デザイン時サポートがあるのは、いずれかの Visual J# プロジェクト テンプレートを使って作成された新しいアプリケーションだけです。ただし、Visual J# で有効な新機能を追加することにより、既存のアプリケーションを拡張できます。たとえば、WFC アプリケーションを Windows フォームによって拡張できます。詳細については、WFCUsingWinForm サンプルを参照してください。
- リソース ファイルの形式が変更されています。詳細については、「リソースを使う Visual J++ 6.0 アプリケーションのアップグレード」を参照してください。
- Visual J# を使って、Visual J++ 6.0 で作成されたプロジェクトを開いてアップグレードできます。Visual J++ プロジェクトを Visual J# で開くと、アップグレード ウィザードが起動し、新しいプロジェクトへの移行を簡単に行うことができます。アップグレード ウィザードは、既存のプロジェクト ファイルの追加先となる新しいプロジェクトを作成します。アップグレード中にソース ファイルが変更されることはありません。また、ウィザードは、問題を識別するためにプロジェクトの分析も行います。検出された問題は、新しいプロジェクトに追加されるアップグレード レポートに一覧表示されます。それぞれの問題には、対処方法について説明したドキュメント内のトピックへのリンクが含まれます。

参照

その他の技術情報

Visual J++ 6.0 からのアップグレード

アップグレードの計画

ほとんどの場合、Visual J# では、Visual J++ プロジェクトを簡単な手順でアップグレードできます。ただし、アップグレードを試す前に、そのプロジェクトが Visual J++ で正しくビルドされ、動作することを確認する必要があります。また、アップグレードの前に、Visual J++ プロジェクトのアクティブな構成をリリース バージョンに設定し、アプリケーションが動作することを確認する必要もあります。これにより、アップグレードが簡単になります。

Visual J++ の機能の中には、Visual J# でサポートされていない機能や、Visual J# では互換性に問題がある機能があります。アプリケーションを Visual J# で正しく実行するために必要な作業を決定するには、「クラス ライブラリのサポート」および「言語サポート」を参照してください。「Visual J++ 6.0 プロジェクトからのアップグレードに関する一般的な問題」には、アプリケーションのアップグレード時に発生する可能性のある問題の一覧が示されています。

アップグレード ウィザードは、ユーザーが入力した情報とプロジェクト ファイルの分析結果に基づいて、プロジェクトのプロパティ、および COM コンポーネントと .NET コンポーネントの参照を設定します。大規模なプロジェクトでは、分析のために、アップグレードされたプロジェクトをビルドする場合と同じくらいの時間が必要になることがあります。ウィザードでは、[完了] ボタンをクリックして分析の手順を省略できますが、分析を省略しないことをお勧めします。ウィザードが作成するアップグレード レポートには、アプリケーションを Visual J# で正しく実行するための問題の多くが記録されます。問題の一覧は、アップグレードを成功させるために必要な作業量を決定するときに役立ちます。

参照

その他の技術情報

Visual J++ 6.0 からのアップグレード

Visual J# アップグレード ウィザード

メモ：
使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

Visual J# には、Visual J++ 6.0 から Visual J# へのプロジェクトのアップグレードを支援するアップグレード ウィザードが用意されています。

アップグレード ウィザードを起動するには、Visual J# で Visual J++ 6.0 のソリューション ファイル (.sln) またはプロジェクト ファイル (.vjp) を開きます。Visual J++ ソリューションに複数のプロジェクトが含まれている場合は、プロジェクトごとにウィザードが起動します。

次に、ウィザードは、Visual J# でアプリケーションをアップグレードして実行する場合の問題を識別するための一連の手順を実行します。検出された問題は、アップグレード レポートに表示されます。詳細については、「Visual J# のアップグレード レポート」を参照してください。

ウィザードは、元のプロジェクトと同じ場所に新しいプロジェクトを作成し、Visual J++ プロジェクトのソース ファイルを新しいプロジェクトに追加します。ソース コードは変更されません。ウィザードは、ソース ファイルを分析して、プロジェクトを Visual J# にアップグレードするときに問題があるかどうかを判断します。分析の結果は、アップグレードされたプロジェクトに適切なプロジェクト プロパティを設定するために使用されます。分析中に検出されたサポートされていない機能の問題や互換性の問題は、すべてアップグレード レポートに追加されます。

ウィザードは、プロジェクトで使用されていることが検出された COM コンポーネントと .NET Framework コンポーネントに対し、参照を追加しようと試みます。プロジェクト内のリソース ファイル (.resources および .properties) も、ウィザードによって .NET Framework リソース ファイル (.resx) 形式にアップグレードされます。

ウィザードの最初のページでは、次のいずれかを選択できます。

[キャンセル]

Visual J# へのアップグレードをキャンセルします。

[次へ]

ウィザードの次のページに移動します。

[完了]

プロジェクト ファイルを分析せずに、プロジェクトをすぐにアップグレードします。アップグレードしたプロジェクトをビルドして実行するには、新しいプロジェクト プロパティとソース ファイルに、必要な変更を手動で加える必要があります。

ウィザードの使い方のヘルプを表示するには、ウィザードが表示されているときに **F1** キーを押します。

このセクションの内容

アップグレード ウィザード : [プロジェクトの種類の選択]

ウィザードを使ってプロジェクトの種類を選択する方法の詳細を示します。

アップグレード ウィザード : [参照をプロジェクトに追加する]

ウィザードを使ってプロジェクト参照を追加する方法の詳細を示します。

参照

処理手順

Visual J++ 6.0 プロジェクトのアップグレード
リソースを使う Visual J++ 6.0 アプリケーションのアップグレード

関連項目

Visual J# のアップグレード レポート
Visual J# のアップグレード リファレンス

アップグレード ウィザード : [プロジェクトの種類の選択]

メモ :
使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

Visual J# アップグレード ウィザードは、Visual J++ プロジェクトをどの種類のプロジェクトにアップグレードするかの決定を試みます。複数の種類のプロジェクトにアップグレードできる場合は、作成するプロジェクトの種類を選択できます。

このウィザード ページでは、プロジェクトの種類を選択し、[次へ] をクリックしてウィザードを継続できます。すぐにアップグレードを実行して、ウィザードの残りの手順を省略するには、[完了] をクリックします。

選択できるプロジェクトの種類を次に示します。

コンソール アプリケーション

Visual J++ プロジェクトが、Jview.exe ツールを使ってコマンド ライン アプリケーションとして実行するように設計されていた場合は、Visual J# でコンソール アプリケーションにアップグレードする必要があります。別のプロジェクトでこのプロジェクト内のクラスを使う場合は、新しいプロジェクトをクラス ライブラリとして作成することを検討する必要があります。たとえば、クラス ライブラリやコントロールを含むプロジェクトをアップグレードする場合は、クラス ライブラリを選択する必要があります。Visual Studio では、他のプロジェクトで参照される出力を持つことができるのは、クラス ライブラリ プロジェクトだけです。

Windows アプリケーション

Visual J++ プロジェクトが、WFC (Windows Foundation Classes) または AWT を使ってアプリケーションとして作成されていた場合は、Visual J# で Windows アプリケーションにアップグレードする必要があります。別のプロジェクトでこのプロジェクト内のクラスを使う場合は、新しいプロジェクトをクラス ライブラリとして作成することを検討する必要があります。

クラス ライブラリ

Visual J++ プロジェクトが、そのクラスを他のプロジェクトで使用できるように作成されていた場合、または COM DLL として作成されていた場合は、Visual J# でクラス ライブラリにアップグレードする必要があります。Visual J++ コントロール プロジェクトも、クラス ライブラリ プロジェクトにアップグレードする必要があります。

プロジェクトのアップグレード後にプロジェクトの種類を変更する場合は、Visual J# でプロジェクト プロパティを変更して、プロジェクトの種類を手動で選択できます。詳細については、「Visual J# プロジェクトのプロパティの設定」を参照してください。

参照

関連項目

Visual J# アップグレード ウィザード

アップグレード ウィザード : [参照をプロジェクトに追加する]

Visual J# アップグレード ウィザードは、Visual J++ 6.0 プロジェクト ファイルを分析して、アップグレードされた Visual J# プロジェクトのプロパティの決定と、サポートされていない API の問題や互換性に関する問題の識別を試みます。ウィザードでソースを完全に分析するには、Visual J++ 6.0 プロジェクトで使用されているクラスのうち、プロジェクト自体または Visual J# クラス ライブラリに定義されていないすべてのクラスへの参照が必要となります。Visual J# でサポートされているクラスの一覧については、「クラス ライブラリのサポート」を参照してください。分析を完了するためには、プロジェクトで使用されている COM クラスや COM インターフェイスを記述する COM タイプ ライブラリへの参照も必要です。

プロジェクトに外部参照が設定されていて、このウィザード ページでそれらの参照を追加しないように選択した場合でも、ウィザードによって Visual J++ 6.0 プロジェクトを Visual J# にアップグレードできます。ただし、アップグレードされたプロジェクトをビルドして実行するには、プロジェクトへの外部参照を後で追加することが必要な場合があります。

Visual J++ プロジェクトに CLASSPATH 設定が定義されている場合は、このウィザード ページに設定内容が表示されます。これで、外部参照の場所を特定できます。

ork[追加] をクリックして、参照リストに追加する 1 つ以上の .NET Framework コンポーネント (.NET Framework クラス ライブラリまたはアセンブリとも呼ばれます) または COM タイプ ライブラリを選択します。リストから項目を削除するには、[削除] を使います。

参照リストに追加した .NET Framework コンポーネントと COM コンポーネントは、アップグレードされたプロジェクトの参照リストに追加されます。

アップグレードの前に、既知のすべての参照をリストに追加することをお勧めします。参照リストに項目を追加しない場合、アップグレード分析では、プロジェクト内にある未解決の外部参照の識別が次の方法で試みられます。

- COM 参照は、レジストリから検索されます。
- クラス参照は、Visual J++ プロジェクトで定義されている CLASSPATH から検索されます。

Visual J++ プロジェクトで別のライブラリやプロジェクトのクラスを使う場合は、それらのクラスへの参照を追加する必要があります。Visual J++ 6.0 では、プロジェクトの CLASSPATH 設定を使って参照先クラスが検索されます。参照先クラスのいずれかが Visual J# クラス ライブラリに含まれていない場合は、そのクラスの実装を含むクラス ライブラリ (DLL) への参照を追加する必要があります。

Visual J++ プロジェクトで、COM DLL またはタイプ ライブラリに定義されているクラスやインターフェイスが使用されている場合は、その DLL またはタイプ ライブラリ ファイル (.tlb) を参照リストに追加できます。プロジェクト内のクラスが、外部で定義されたクラスやインターフェイスを拡張または実装している場合は、それらのクラスやインターフェイスへの参照が使用できないと、クラスの分析が不完全になることがあります。

ソース ファイルがない場合は、Visual J# バイナリ変換ツールを使って、クラス ファイル (.class) またはアーカイブ ファイル (.cab、.jar、または .zip) をクラス ライブラリに変換できます。

参照

関連項目

Visual J# アップグレード ウィザード

Visual J# バイナリ変換ツール (Java 言語のバイトコードから MSIL への変換用)

その他の技術情報

クラス ライブラリのサポート

Visual J# のアップグレード レポート

Visual J# アップグレード ウィザードは、Visual J++ 6.0 から Visual J# にアップグレードされるすべてのプロジェクトについてのアップグレード レポートを作成します。このレポートは、アップグレードされたプロジェクトに追加されます。既定では、ウィザードの終了時にアップグレード レポートが開かれます。アップグレード レポートには、アップグレード プロセスに関する情報と、プロジェクトの実行前に対処する必要のある問題の一覧が表示されます。

アップグレード レポートは HTML 形式です。ソリューション エクスプローラ内でファイルをダブルクリックすると Visual Studio 開発環境内で表示できます。また、[ファイル] メニューの [ブラウザの選択] をクリックして、外部のブラウザで表示することもできます。

レポートには、次の 3 つのセクションがあります。

- 最初の部分には、アップグレードされたプロジェクトの場所やアップグレードの時刻など、アップグレードに関する一般情報が含まれます。
- 2 番目の部分には、アップグレード時に検出された問題の一覧が含まれます。問題の各カテゴリのセクションを展開することにより、問題の影響やプロジェクトでの発生回数など、問題の詳細を参照できます。さらに展開していくと、問題が検出されたファイルなど、関連する詳細情報を参照できます。それぞれのカテゴリと問題の説明は、問題の対処方法に関する詳細情報が記載されたヘルプ トピックにリンクしています。
- 最後の部分には、アップグレードされたプロジェクトに追加された参照のリストなど、アップグレードに関する追加情報が含まれます。

参照

関連項目

Visual J# アップグレード ウィザード

Visual J# のアップグレード リファレンス

その他の技術情報

Visual J++ 6.0 からのアップグレード

方法：Visual J++ 6.0 プロジェクトのアップグレード

メモ：
使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

Visual J++ 6.0 のソリューションとプロジェクトは、Visual J# アップグレード ウィザードを使って簡単に Visual J# にアップグレードできます。Visual Studio で Visual J++ ソリューションを開くと、アップグレード ウィザードを使って各プロジェクトをアップグレードするかどうかを確認するメッセージが表示されます。アップグレード レポートが生成され、アップグレードされた各プロジェクトに追加されます。このレポートには、プロジェクト ファイルの分析中にアップグレード ウィザードによって検出されたアップグレードの問題の一覧が表示されます。

メモ：
アップグレード中にプロジェクト内のソース ファイルが変更されることはありません。

Visual J++ 6.0 でソリューションを開いている場合は、アップグレードを開始する前にソリューションを閉じる必要があります。プロジェクトがソース管理の対象になっている場合は、ソリューション ファイルに書き込みアクセス権が必要です。

Visual J++ 6.0 プロジェクトを Visual J# にアップグレードするには

1. Visual J++ 6.0 でプロジェクトを開きます。アクティブな構成をリリース バージョンに設定します。プロジェクトを作成します。アプリケーションを実行し、エラーがないことを確認します。
2. Visual J++ 6.0 を終了し、Visual J# がインストールされている Visual Studio を起動します。[ファイル] メニューの [開く] をポイントし、[プロジェクト] をクリックします。
3. [プロジェクトを開く] ダイアログ ボックスで Visual J++ プロジェクトの場所を参照し、プロジェクト ファイル (.vjp) またはソリューション ファイル (.sln) を選択します。これにより、アップグレード ウィザードが起動します。
4. ウィザードの最初のページで [完了] をクリックすると、プロジェクト分析を省略できます。ただし、ウィザードのすべての手順を継続することをお勧めします。

ヒント
詳細については、ウィザードの実行中に F1 キーを押してください。

5. ウィザードの 2 ページ目で、使用できるプロジェクトの種類を選択してプロジェクトをアップグレードします。

ヒント
使用できるオプションの説明については、ページの右下隅にある [詳細] リンクをクリックしてください。

6. ウィザードの 3 ページ目で、プロジェクトで使用されるクラスを含む COM ライブラリと .NET Framework コンポーネントへの参照を追加します。

ヒント
参照の追加の詳細については、ページの右下隅にある [詳細] リンクをクリックしてください。

7. [次へ] をクリックして、アップグレードの問題を検出するためのプロジェクト ファイルの分析を開始します。この処理には、プロジェクトのサイズに応じて数分かかることがあります。
8. 分析が完了すると、ウィザードは次のページを表示します。[完了] をクリックしてプロジェクトのアップグレードを終了します。ソリューション エクスプローラに、アップグレードされたプロジェクトが表示されます。
9. 既定では、アップグレード レポートが開かれます。アップグレード レポートが開かない場合は、ソリューション エクスプローラで _UpgradeReport.htm ノードをダブルクリックして、アップグレード レポートを表示します。

10. アップグレード レポートの内容を確認します。ビルド エラーとランタイム エラーの原因となる問題をすべて修正します。

ヒント
問題の説明をクリックすると、その問題に関するヘルプ トピックが表示されます。

11. F5 キーを押して、アップグレードされたプロジェクトをビルドして実行します。

Visual J# では、Visual J++ 6.0 でデバッグ バージョンとリリース バージョンのどちらに対しても構成できるビルド設定の一部が、共通プロパティになっています。次に例を示します。

- [出力の種類] は、Windows アプリケーション、コンソール アプリケーション、またはクラス ライブラリを示します。
- [スタートアップ オブジェクト] は、アプリケーションの起動時に **main** メソッドが呼び出されるクラスを示します。

これらのプロパティの値は、Visual J# へのアップグレード時に、Visual J++ プロジェクトのリリース バージョンの設定から取得されます。アップグレードされたプロジェクトに関する一般的な問題のヘルプについては、「Visual J++ 6.0 プロジェクトからのアップグレードに関する一般的な問題」を参照してください。

アップグレード プロセスでは、バックアップ用の Visual J++ ソリューション ファイルに .old という拡張子が追加されます。たとえば、vj6App.sln は vj6App.sln.old という名前に変更されます。アップグレード プロセスで作成された新しいソリューション ファイルの名前は、元の Visual J++ ソリューション ファイルと同じ名前 (vj6App.sln) になります。新しいプロジェクト ファイルには、.vjsproj という拡張子が追加されます。

元の (バックアップ) Visual J++ ソリューションを Visual J++ で使うには

1. アップグレードされた新しいソリューション ファイルの名前を変更します (vj6App.new.sln など)。
2. 拡張子 .suo が付いたファイルの名前を変更します。たとえば、vj6App.suo を vj6App.new.suo に変更します。

メモ :
Windows では、拡張子 .suo が付いたファイルは既定で隠しファイルになります。

3. 元のソリューション ファイルに追加した拡張子 .sln.old から .old を削除します。たとえば、vj6App.sln.old を vj6App.sln に変更します。
4. 手順 3. のソリューション ファイルを Visual J++ で開きます。

メモ :
アップグレードのすべての問題をウィザードで検出できるとは限りません。特に、必要なすべての参照がウィザードの 3 ページ目で指定されなかった場合は、一部の問題を検出できない可能性があります。

参照

処理手順

方法 : リソースを使う Visual J++ 6.0 アプリケーションのアップグレード

関連項目

Visual J# アップグレード ウィザード

Visual J# のアップグレード レポート

その他の技術情報

Visual J++ 6.0 からのアップグレード

方法：リソースを使う Visual J++ 6.0 アプリケーションのアップグレード

メモ：
使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

共通言語ランタイムでリソースを使うために、既存の Visual J++ リソースをマネージ アセンブリに埋め込んだり、マネージ アセンブリにリンクしたりするには、リソース ファイルを .NET Framework でサポートされている形式に変換する必要があります。

リソース ファイルの拡張子は、.NET Framework でも Visual J++ 6.0 でも .resources ですが、実際には 2 つのファイル形式は大きく異なるため、互いに置き換えることはできません。

リソースの変換

Visual J# アップグレード ウィザードは、Visual J++ プロジェクトをアップグレードするときに、そのプロジェクトに関連付けられている Visual J++ リソース ファイル (.resources および .properties) を自動的に .NET Framework リソース ファイル (.resx) に変換します。後で Visual J# でプロジェクトをビルドすると、.resx ファイルが .NET Framework リソース ファイル (.resources) に変換され、プロジェクトの出力に埋め込まれます。

.resources または .properties 以外のファイル形式 (.bmp や .gif など) のリソースは、アップグレード ウィザードでは自動的に処理されません。これらのリソースは、手動で .resx ファイル形式に変換し、アップグレードされたプロジェクトに追加する必要があります。

Visual J++ 6.0 プロジェクト リソースを手動で .NET Framework リソースに変換する手順を次に示します。

リソースを手動で変換するには

1. Visual J++ プロジェクトで使用されているすべてのリソース ファイルを識別します。
2. Visual J# のサンプルに含まれている Vjsresgen.exe ツールを使って、これらのリソース ファイルを .NET Framework リソース ファイル (.resx) に変換します。
3. アップグレードされたプロジェクトに、生成した .resx ファイルを追加します。
4. コマンド プロンプトでプロジェクトをコンパイルするには、.NET Framework ツールのリソース ファイル ジェネレータ (Resgen.exe) を使って、.resx ファイル形式のリソースを .resources ファイル形式に変換します。これで、マネージ アセンブリにリソースを埋め込んだり、リンクしたりする準備ができます。
5. マネージ アセンブリにリソースを埋め込むには、Visual J# コンパイラの /resource オプションを指定します。アプリケーションが .class 形式でしか使用できない場合は、Visual J# バイナリ変換ツールを使って **/linkresource** オプションを指定します。

Visual J++ 6.0 とのリソースの互換性の問題

Visual J++ 6.0 では、すべてのリソースがクラスに関連付けられます。クラスに関連付けられているリソースは、そのクラスが含まれているアセンブリに埋め込むか、またはリンクする必要があります。

- Visual J# では、元の Visual J++ リソース ファイルが変更された場合は、.NET Framework リソース ファイルを再作成してアセンブリに再び追加する必要があります。
- リソースを動的に生成して利用することはサポートされていません。
- /win32res コンパイラ オプションを使って追加されたリソース ファイルには、**Class.getResource** メソッドまたは **Class.getResourceAsStream** メソッドではアクセスできません。
- クラスに関連付けられているリソースは、そのクラスが含まれているアセンブリに埋め込むか、またはリンクする必要があります。単一のリソースが複数のクラスで共有される場合は、それらのクラスが組み込まれるアセンブリごとに、リソースを埋め込むかリンクする必要があります。

マネージ アセンブリへのリソースの埋め込みまたはリンクに使う Visual J# コンパイラ オプションの詳細については、「Visual J# コンパイラ オプション」を参照してください。Visual J# バイナリ変換ツール (JbImp.exe) のオプションの詳細については、「Visual J# バイナリ変換ツール (Java 言語のバイトコードから MSIL への変換用)」を参照してください。

Resgen.exe の詳細については、「リソース ファイル ジェネレータ (Resgen.exe)」を参照してください。

参照

処理手順

方法 : Visual J++ 6.0 プロジェクトのアップグレード

関連項目

Visual J# アプリケーションでのリソースの使用

Visual J# アップグレード ウィザード

その他の技術情報

Visual J++ 6.0 からのアップグレード

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

このセクションでは、Visual J++® 6.0 を使って開発された Java 言語/COM コンポーネントを Visual J# でコンパイル、デバッグ、および実行する方法について説明します。

このセクションの内容

概要 : Java 言語/COM コンポーネントのアップグレード

Java 言語/COM 相互運用機能を使った Visual J++ 6.0 プロジェクトの概要とアップグレード方法について説明します。

背景情報 : Visual J++ 6.0 での COM 相互運用機能

Java 言語から COM コンポーネントにアクセスする方法と、COM から Java 言語コンポーネントにアクセスする方法について説明します。

背景情報 : 共通言語ランタイムでの COM 相互運用機能

マネージ コードから COM コンポーネントにアクセスする方法と、COM からマネージ コンポーネントにアクセスする方法について説明します。

COM コンポーネントにアクセスする Visual J++ 6.0 アプリケーションのアップグレード

COM コンポーネントにアクセスする Visual J++ 6.0 アプリケーションを Visual J# でコンパイルして実行する方法について、詳細な手順と背景情報を説明します。

COM に公開される Visual J++ 6.0 コンポーネントのアップグレード

COM コンポーネントからアクセスされる Visual J++ 6.0 アプリケーションを Visual J# でコンパイルして実行する方法について、詳細な手順と背景情報を説明します。

Visual J# でサポートされていないシナリオ

Visual J# でサポートされていない Java 言語/COM のシナリオについて説明します。

既知の問題と問題回避

Java 言語/COM のシナリオでの問題回避について説明します。

関連するセクション

アンマネージ コードとの相互運用

共通言語ランタイムによって提供される相互運用サービスについて説明します。

Visual J++ 6.0 からのアップグレード

Visual J++ 6.0 プロジェクトを Visual J# にアップグレードする場合の推奨事項と手順について説明します。

COMCallsJava サンプル

COM DLL として公開される Visual J++ 6.0 の Java 言語コンポーネントをアップグレードする方法について説明します。

JavaCallsCOM サンプル

Visual J++ 6.0 の Java 言語/COM アプリケーションを Visual J# にアップグレードする方法について説明します。

Visual J#

Visual J# ドキュメントのさまざまなトピックへのリンクの一覧です。

概要：Java 言語/COM コンポーネントのアップグレード

Visual J# では、マネージ アプリケーションと COM コンポーネントとの間の相互運用性がサポートされています。このサポートには、既存の Visual J++ 6.0 コンポーネントと、.NET Framework の相互運用セマンティクスを使って記述された新しいコードをコンパイルして実行する機能が含まれています。

したがって、マネージ アプリケーションと COM コンポーネントは、相互のサービスにアクセスできます。つまり、開発者は、Java 言語の構文と COM の利点を組み合わせ、COM と互換性のある任意の言語で記述された任意のプラットフォーム上のオペレーティング システム コンポーネント、アプリケーション、およびサービスとやり取りするアプリケーションを開発できます。

新しいアプリケーションで COM コンポーネントとマネージ コードの相互運用を行う場合は、共通言語ランタイムおよびツールに用意されている .NET Framework の相互運用セマンティクスを使う方法が推奨されています。COM 相互運用のための共通言語ランタイムのセマンティクスの使い方の詳細については、「アンマネージ コードとの相互運用」を参照してください。

Visual J# では、Visual J++ 6.0 の Java 言語/COM コンポーネントのコンパイルと実行がサポートされています。具体的には、次の情報が含まれています。

- Visual J++ 6.0 と Microsoft® SDK for Java で使用できる Java 言語/COM 相互運用のほとんどのシナリオおよび機能のサポート。
- Visual J++ 6.0 の Java 言語/COM アプリケーションに対するソース互換性。ソース コードは、ほとんどまたは一切変更せずにコンパイルして実行できます。既存の Visual J++ 6.0 アプリケーションで JActiveX™ ツールによって生成されたラッパーや、これらのラッパーの @com ディレクティブをコンパイルして実行する機能も含まれています。

メモ：

Visual J# では、Visual J++ 6.0 の既存の Java 言語/COM アプリケーションをすべてコンパイルして実行できるような、完全なソース互換性が保証されているわけではありません。詳細については、「Visual J# でサポートされていないシナリオ」を参照してください。

参照

その他の技術情報

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

背景情報 : Visual J++ 6.0 での COM 相互運用機能

Visual J++ 6.0 の Java 言語/COM 相互運用機能では、2 つの主要な使用シナリオがサポートされていました。

このセクションの内容

Java 言語からの COM コンポーネントへのアクセス

Java 呼び出し可能ラッパー (JCW: Java Callable Wrapper) を使って、Java 言語コードから COM コンポーネントにアクセスする方法について説明します。

COM からの Java 言語コンポーネントへのアクセス

Visual J++ 6.0 で、Java 言語クラスを COM コンポーネントとしてコンパイルして登録するためのサポートについて説明します。

関連するセクション

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

Java 言語/COM 相互運用機能を使う Visual J++ 6.0 のプロジェクトをアップグレードする場合の考慮点について説明します。

Java 言語からの COM コンポーネントへのアクセス

COM コンポーネントにアクセスするには、COM タイプ ライブラリ (.tlb ファイル) またはタイプ ライブラリが埋め込まれた COM DLL に対して JActiveX ツールを実行します。Visual J++ 6.0 および Microsoft SDK for Java に付属の JActiveX ツールは、タイプ ライブラリ内の各コクラスと COM インターフェイスに対応するラッパーを生成します。Java 呼び出し可能ラッパー (JCW: Java Callable Wrapper) と呼ばれるこれらのラッパーは、実際の COM オブジェクト用のプロキシで、Java 言語コンパイラ (jvc.exe) で認識される **@com** ディレクティブを含みます。次に、これらのラッパーをアプリケーションにインポートし、アプリケーション コードと共にコンパイルします。ラッパー内の **@com** ディレクティブは、コンパイルされて .class ファイルの属性になり、MSJVM (Microsoft Java Virtual Machine) によって実行時に解釈されます。

COM オブジェクトのインスタンスを作成するには、適切なラッパーに対して **new** を呼び出します。その後、オブジェクトをインターフェイスにキャストして、そのインターフェイスのメソッドを呼び出すことができます。MSJVM では、生成されたラッパーの **@com** ディレクティブを使って、次の処理を行います。

- COM コクラスで **CoCreateInstance** を探して呼び出す。
- ラッパーに対するメソッド呼び出しを変換して、実際の COM オブジェクトのメソッドを呼び出す。
- COM オブジェクトへの参照を管理する。
- COM と Java 言語との間でパラメータをマーシャリングする。

Java 言語の開発者は、他の Java 言語オブジェクトを使う場合と同様に COM オブジェクトを使うことになります。通常の COM 開発者のように、詳細な処理を行う必要はありません。

また、Java 言語で ActiveX コントロールをホストすることもできます。

参照

その他の技術情報

背景情報 : Visual J++ 6.0 での COM 相互運用機能

COM からの Java 言語コンポーネントへのアクセス

Visual J++ 6.0 では、Java 言語クラスを COM DLL や ActiveX コントロール (OCX) にコンパイルするプロジェクトがサポートされています。このような COM DLL または ActiveX OCX は、登録されて Visual Basic 6.0 や Visual C++ 6.0 などのツールから使用されます。また、Visual J++ 6.0 および Microsoft SDK for Java に付属する VJReg や JavaReg などのコマンド ライン ツールを使って、**@com.register** 属性を持つ Java 言語クラスを COM または ActiveX コントロールとして登録し、Visual Basic 6.0 や Visual C++ 6.0 からアクセスできるようにすることもできます。

Java 言語/COM コンポーネントのコンパイルと実行の詳細については、Visual J++ 6.0 ドキュメント (MSDN) または Microsoft SDK for Java の関連ドキュメントを参照してください。

参照

その他の技術情報

背景情報 : Visual J++ 6.0 での COM 相互運用機能

http://www.microsoft.com/java/resource/java_com2.htm

http://www.microsoft.com/java/resource/java_com.htm

背景情報：共通言語ランタイムでの COM 相互運用機能

.NET Framework の COM 相互運用セマンティクスを使って、マネージ クライアントから COM コンポーネントにアクセスしたり、COM クライアントからマネージ サーバーにアクセスしたりできます。

このセクションの内容

マネージ コードからの COM コンポーネントへのアクセス

ランタイム呼び出し可能ラッパー (RCW: Runtime Callable Wrapper) を使って、マネージ コードから COM コンポーネントにアクセスする方法について説明します。

COM からのマネージ コンポーネントへのアクセス

Regasm.exe ツールを使って、マネージ DLL を COM コンポーネントとして登録する方法について説明します。

関連するセクション

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

Java 言語/COM 相互運用機能を使う Visual J++ 6.0 のプロジェクトをアップグレードする場合の考慮点について説明します。

方法：マネージ コードからの COM コンポーネントへのアクセス

メモ：

使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

COM タイプ ライブラリまたはタイプ ライブラリが埋め込まれている COM DLL に対してタイプ ライブラリ インポータ ツール (Tlbimp.exe) を実行すると、COM コンポーネントのマネージ ラッパーが生成されます。Tlbimp.exe ツールは、.NET Framework SDK および Visual Studio に付属しています。生成されたラッパーは、COM DLL 内のクラスにアクセスする必要のある任意のマネージ アプリケーションで参照できます。

これらのラッパーは、ランタイム呼び出し可能ラッパー (RCW: Runtime Callable Wrapper) と呼ばれ、実際の COM オブジェクトに対するプロキシとして機能します。RCW には、.NET Framework の相互運用属性が含まれます。これらの属性は、実行時に実際の COM オブジェクトのインスタンスを作成するために、共通言語ランタイムによって解釈されます。

マネージ コードから COM コンポーネントにアクセスするには

1. Visual J# アプリケーションからアクセスしようとしている COM DLL について、COM タイプ ライブラリからマネージ ラッパーを生成します。マネージ ラッパーの生成には、次のように Tlbimp.exe を使います。

   ```
   tlbimp /out:FB.dll My.tlb
   ```

2. アプリケーションで、生成されたラッパーを参照します。その後、適切なラッパーに対して **new** を呼び出すことにより、COM コンポーネントにアクセスできます。ソースをコンパイルするときは、生成されたラッパーを次のように参照します。

   ```
   vjc /reference:FB.dll application_sources
   ```

3. COM オブジェクトのインスタンスを作成するには、適切なコクラスのマネージ ラッパーに対して **new** を使います。オブジェクトを適切なインターフェイスにキャストすると、そのインターフェイスのメソッドを呼び出すことができます。共通言語ランタイムは、ラッパーに含まれている相互運用属性を使って、COM コクラスの **CoCreateInstance** の検索と呼び出し、ラッパーに対するメソッド呼び出しから実際の COM オブジェクトに対する呼び出しへの変換、COM オブジェクトへの参照の管理、および COM 型のパラメータの .NET Framework 型へのマーシャリングを行います。

   このサポートには、Windows フォームを ActiveX コントロールのホストにする機能も含まれています。

参照

その他の技術情報

背景情報 : 共通言語ランタイムでの COM 相互運用機能

方法：COM からのマネージ コンポーネントへのアクセス

メモ：

使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

アセンブリ登録ツール (Regasm.exe) を使うと、マネージ DLL を COM コンポーネントとして登録できます。このため、Visual Basic や Visual C++ などのツールから、COM DLL または ActiveX コントロールとしてマネージ コンポーネントにアクセスできます。さらに、タイプ ライブラリ エクスポータ ツール (Tlbexp.exe) や Regasm の /tlb オプションを使って、マネージ コンポーネントのタイプ ライブラリを生成することもできます。これらのツールは、.NET Framework SDK および Visual Studio に付属しています。

COM からマネージ コンポーネントにアクセスする基本手順

1. .NET Framework ツール Regasm.exe を使って、コンポーネントを COM サーバーとして登録します。

```
Regasm /tlb:mytlb.tlb /codebase managedobj.dll
```

2. COM コンポーネントとして登録されたマネージ クラスには、アンマネージ クライアントからアクセスできます。

.NET Framework のセマンティクスを使った COM 相互運用機能の詳細については、「アンマネージ コードとの相互運用」を参照してください。

参照

その他の技術情報

背景情報 : 共通言語ランタイムでの COM 相互運用機能

COM コンポーネントにアクセスする Visual J++ 6.0 アプリケーションのアップグレード

Visual J# では、COM コンポーネントにアクセスする Visual J++ 6.0 の Java 言語/COM アプリケーションをコンパイルして実行できます。

このセクションの内容

方法 : COM コンポーネントにアクセスする Visual J++ アプリケーションのアップグレード

COM コンポーネントにアクセスする Visual J++ 6.0 のアプリケーションを Visual J# にアップグレードする手順について説明します。

デザインの概要 : COM コンポーネントへのアクセス

COM コンポーネントにアクセスする Visual J++ 6.0 アプリケーションをアップグレードして .NET Framework で実行できるようにするためのデザイン機能について説明します。

COM コンポーネントのスレッド モデルの処理

COM オブジェクトのスレッド モデルの処理方法について、MSVJM (Microsoft Java Virtual Machine) と .NET Framework 共通言語ランタイムとの違いを説明します。

関連するセクション

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

Java 言語/COM 相互運用機能を使う Visual J++ 6.0 のプロジェクトをアップグレードする場合の考慮点について説明します。

方法：COM コンポーネントにアクセスする Visual J++ アプリケーションのアップグレード

コマンド プロンプトから手動で Java 言語/COM コンポーネントをアップグレードする手順を次に示します。Visual Studio で Visual J++ プロジェクトをアップグレードするときは、Visual J# アップグレード ウィザードによってこれらの手順の多くが自動化されます。

Visual J++ アプリケーションを Visual J# にアップグレードするには

1. Tlbimp.exe ツールを使って、COM コンポーネントのタイプ ライブラリに対するマネージ ラッパーを生成します。通常、この操作は次のコマンド ラインを使って行います。

   ```
   tlbimp.exe /out:managed_WrapperDLL COMComponent.tlb
   ```

2. Tlbimp で生成されたマネージ ラッパーを参照して、Java 言語アプリケーションを JActiveX ラッパーと共にコンパイルします。コンパイルには、Visual J# コンパイラ vjc.exe を使います。通常、この操作は次のコマンド ラインを使って行います。

   ```
   vjc ApplicationSources JActiveXWrapperSources /r:managed_WrapperDLL
   ```

3. アプリケーションを実行するには、生成された .exe ファイルを実行します。

4. .NET Framework に用意されているデバッグ ツールの CorDbg.exe や DbgClr.exe、または Visual Studio デバッガを使って、アプリケーションをデバッグできます。これらのツールの詳細については、「アンマネージ コードとの相互運用」および「デバッグ」を参照してください。

   Tlbimp で生成されるマネージ ラッパーに、JActiveX ラッパーとの互換性はありません。したがって、JActiveX ラッパーを破棄し、アプリケーションにマネージ ラッパーを直接インポートする場合は、アプリケーションでソース コードを大幅に変更する必要があります。たとえば、Tlbimp で生成されたマネージ ラッパーのメソッドは、パラメータおよび戻り値として .NET Framework 型を受け入れますが、JActiveX ツールで生成されたラッパーのメソッドは、パラメータおよび戻り値として Java 言語型を受け入れます。この他にも、これら 2 つのラッパーでは、プロパティとイベントの公開方法、HRESULT エラーのレポート方法、およびラッパーが実装するインターフェイスに違いがあります。

   Visual J++ 6.0 の Java 言語/COM アプリケーションを Visual J# にアップグレードするときに、JActiveX で生成されたラッパーを手動で編集して .NET Framework COM 相互運用機能の属性や API を追加すると、予期しない問題が実行時に発生する可能性があります。Visual J++ 6.0 の **@com** や **@dll** などの属性と .NET Framework COM 相互運用機能の属性を混在させることはサポートされていません。Visual J++ 6.0 の Java 言語/COM アプリケーションをアップグレードする場合でも、Visual J# で新しいアプリケーションを記述する場合でも、これらの属性を混在させないようにすることを強く推奨します。

参照

その他の技術情報

COM コンポーネントにアクセスする Visual J++ 6.0 アプリケーションのアップグレード
Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

デザインの概要：COM コンポーネントのアクセス

Visual J# の基本的な目標は、ソースの互換性を維持しながら、共通言語ランタイムに用意されている相互運用機能のサポートを使うことでした。このため、Visual J++ 6.0 の Java 言語/COM コンポーネントは、JActiveX ラッパーを使った場合でも、Tlbimp によって生成されたマネージ ラッパーを使った場合でも、Visual J# でコンパイルして実行できます。

Visual J++ 6.0 アプリケーションをアップグレードするには、vjc.exe を使って、JActiveX で生成されたラッパーとアプリケーションをコンパイルします。このとき、Tlbimp で生成されたマネージ ラッパー アセンブリを参照します。Visual J# コンパイラ (vjc.exe) は、@com 対応のコンパイラです。vjc.exe は、Tlbimp で生成されたマネージ ラッパーを使う JActiveX ラッパーの実装をコンパイル時に生成します。コンパイラは、各 JActiveX ラッパーが対応するマネージ ラッパーに接続し、そのマネージ ラッパーへの参照を保持するようなコードを出力します。JActiveX ラッパーの新しいインスタンスが作成されるたびに、対応するマネージ ラッパーの新しいインスタンスも作成されます。これにより、.NET Framework 相互運用インフラストラクチャによって実行される COM コクラスが作成されます。

JActiveX ラッパーのメンバは、マネージ ラッパーのメンバに対応付けられます。その結果、JActiveX ラッパーのメソッドが呼び出されるたびに、マネージ ラッパーのメソッドが呼び出されます。その後、共通言語ランタイムは、対応する COM コクラスのメソッドを呼び出します。この対応付けは、JActiveX ラッパーの @com 属性を使ってコンパイル時に行われます。

JActiveX ラッパーの実装は、2 つのラッパー間の違いをシームレスに隠ぺいするためにも使用されます。JActiveX ラッパーの実装は、マネージ ラッパーのメソッドを呼び出す前に、Java 言語型を .NET Framework 型にマーシャリングします。ランタイムは、これらの型を適切な COM 型にマーシャリングします。メソッドの戻り値に対しては、その逆のマーシャリングも行われます。したがって、これらのアプリケーションでは、JActiveX で生成されたラッパーが引き続き使用されますが、COM オブジェクトのインスタンス作成とメソッド呼び出しには共通言語ランタイムのサポートが使用されます。

この方法の利点は、JActiveX で生成されたラッパーを、アプリケーションのコード内で使い続けることができる点にあります。

Visual J# にアップグレードした後の Java 言語/COM コンポーネント

COM オブジェクトは、Visual J++ 6.0 と同じ com.ms.* クラスとインターフェイスを継承するオブジェクトとして引き続き公開されます。アプリケーションにとってのこれらのラッパーのレイアウト特性とランタイム特性は、Visual J++ 6.0 と同じになります。COM インターフェイスへのキャストおよびインターフェイスのメソッド呼び出しは、Visual J++ 6.0 の場合と同様に動作します。

COM メソッドに渡される Java 言語型は、適切な COM 型にマーシャリングされます。COM メソッドの戻り値は、適切な Java 言語型にマーシャリングされます。2 つの型間の対応付けは、Visual J++ 6.0 での対応付けと同様で、**IUnknown** や **IDispatch** などのインターフェイス、VARIANT、SAFEARRAY、構造体、列挙型などのデータ型、およびユーザー定義のクラスとインターフェイスが含まれています。

JActiveX ラッパーは、COM メソッドからエラー値の HRESULT が返された場合に **com.ms.com.ComFailException** をスローします。

COM イベントのシンクおよびプロパティへのアクセスのセマンティクスは変更されていません。

Java 言語のオブジェクトは、COM インターフェイスを実装できます。この種のオブジェクトが COM に渡された場合、そのインターフェイスのオブジェクトに対する **QueryInterface** の呼び出しでは適切なポインタが返されます。

参照

その他の技術情報

COM コンポーネントにアクセスする Visual J++ 6.0 アプリケーションのアップグレード

COM コンポーネントのスレッド モデルの処理

Visual J++ 6.0 の Java 言語/COM コンポーネントを Visual J# で実行するときは、MSJVM (Microsoft Java Virtual Machine) と .NET Framework 共通言語ランタイムとの間で、COM オブジェクトのスレッド モデルの処理方法に違いがあることに注意してください。Visual J# での COM 相互運用機能のサポートは .NET Framework COM 相互運用機能の上位に階層化されているため、Visual J# の Java 言語/COM アプリケーションは、次のような問題の影響を受ける可能性があります。

- アパートメント境界を越える COM インターフェイスのマーシャリング
- STA オブジェクトに送信されたメッセージの処理

参照

その他の技術情報

COM コンポーネントにアクセスする Visual J++ 6.0 アプリケーションのアップグレード

アパートメント境界を越える COM インターフェイスのマーシャリング

共通言語ランタイムでは、純粋に COM から COM を呼び出すシナリオでアパートメント境界を越えてマーシャリングできない COM インターフェイスを、互換性のないアパートメント モデルのスレッドを越えてマーシャリングできません。つまり、アパートメント境界を越えるマーシャリングができない COM インターフェイスの場合、そのインターフェイスに対応する JActiveX ラッパーのメソッドは、互換性のないアパートメント モデルのスレッドから呼び出すことができません。

メモ :

このシナリオは、MSJVM (Microsoft Java Virtual Machine) ではサポートされていますが、共通言語ランタイムではサポートされていません。MSJVM は、互換性のないアパートメント モデルのスレッドから COM インターフェイスを呼び出すことができ、Java 言語開発者に意識させることなく、インターフェイス ポインタのマーシャリングを処理します。

既定では、共通言語ランタイムのすべてのスレッド (Visual J# にアップグレードされた Visual J++ 6.0 アプリケーションも) は、マルチスレッド アパートメント (MTA: MultiThreaded Apartment) に属します。つまり、既定では、マネージ スレッドはマルチスレッド オブジェクトだけをサポートするように初期化され、同期とマーシャリングについては開発者に任されます。

したがって、この問題は、シングルスレッド アパートメント (STA: Single-Threaded Apartment) COM オブジェクトのホストになる Visual J++ 6.0 アプリケーションをアップグレードするときに重要になります。この種のアプリケーションでは、COM オブジェクトによって実装されている純粋な **IUnknown** (vtable) インターフェイスのメソッドを呼び出そうとすると、例外がスローされます。純粋な **IUnknown** インターフェイス自体は、アパートメントを越えてマーシャリングできません。すべてのスレッドは既定で MTA に属しているため、これらのインターフェイスを呼び出した場合、**com.ms.com.ComFailException** が発生します。この問題を解決するには、次のいずれかの方法を使用し、COM インターフェイスを、アパートメントを越えてマーシャリングできるようにします。

- COM コンポーネントの IDL ファイルから、プロキシ/スタブ DLL を生成して登録します。この作業は、MIDL ツールを使って実行できます。
- デュアル インターフェイスまたはディスパッチ インターフェイスを作成し、タイプ ライブラリを登録して、COM がタイプ ライブラリ ベースでのマーシャリングを行うことができるようにします。この方法は、インターフェイスで使用されている型がオートメーションと互換性がある場合にだけ使用できます。アパートメント境界を越える COM インターフェイスのマーシャリングの詳細については、MSDN ドキュメントを参照してください。
- COM オブジェクトのホストとなっているマネージ スレッドのアパートメント モデルと、そのオブジェクトのスレッド モデルとの互換性を確保します。既定では、スレッドはすべて MTA であるため、STA COM オブジェクトを扱うには、スレッドのアパートメント モデルを STA に設定する必要があります。これは、次の 2 つの方法で行うことができます。
  - エントリ ポイント関数 main で、`/** @attribute System.STAThread () */` 属性を指定します。これにより、アプリケーションのメイン スレッドが STA スレッドになります。次に例を示します。

    ```
    /** @attribute System.STAThread () */
    public static void main(String args[])
    {
       ...
       new STACOMObject();
       ...
    }
    ```

  - .NET Framework クラス **Thread** の **ApartmentState** プロパティを呼び出して、このプロパティを STA にします。この処理を実行できるのは、いずれかの COM コンポーネントの呼び出しが行われる前だけです。詳細については、「ApartmentState プロパティ」を参照してください。

    ```
    System.Threading.Thread.get_CurrentThread().set_ApartmentState(System.Threading.A
    partmentState.STA)
    ```

メモ :

この処理は、ActiveX コントロールのホストになる WFC アプリケーションを、Visual J++ から Visual J# にアップグレードするときに必要です。この場合、メイン スレッドが ActiveX コントロールのホストになるときは、エントリ ポイント関数 main に `/** @attribute System.STAThreadAttribute() */` 属性を追加する必要があり、他のスレッドが ActiveX コントロールのホストになるときは、**ApartmentState** プロパティを使う必要があります。

参照

関連項目

COM コンポーネントのスレッド モデルの処理

STA オブジェクトに送信されたメッセージの処理

シングルスレッド アパートメント (STA: Single-Threaded Apartment) COM オブジェクトのホストになっている STA スレッドがブロックされた場合、通常は共通言語ランタイムが COM オブジェクトに代わって暗黙のポンプを実行します。ただし、次の場合には、共通言語ランタイムによる暗黙のポンプは実行されません。

- スレッドが **Thread.Sleep** の呼び出しでブロックされた場合
- スレッドがプラットフォーム呼び出しまたは J/Direct を通じてマネージ ネイティブ境界を越え、ネイティブ コード内でブロックされた場合

このような場合、STA COM オブジェクトにメッセージは配信されません。

次のサンプル コードに示すように、**com.ms.win32** パッケージのクラスを使ってスレッドに明示的なメッセージ ポンプを実装すると、この問題を解決できます。

```
com.ms.win32.MSG msg = new com.ms.win32.MSG();
while (com.ms.win32.User32.GetMessage(msg, 0, 0, 0))
{
   com.ms.win32.User32.TranslateMessage(msg);
   com.ms.win32.User32.DispatchMessage(msg);
}
```

参照

関連項目

COM コンポーネントのスレッド モデルの処理

方法：COM に公開される Visual J++ 6.0 コンポーネントのアップグレード

メモ：

使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

COM クライアントからアクセスされる Visual J++ 6.0 の Java 言語/COM コンポーネントは、コンパイルして実行できます。コマンド プロンプトから手動で Java 言語/COM コンポーネントをアップグレードする手順を以下に示します。Visual Studio で Visual J++ プロジェクトをアップグレードする場合、Visual J# アップグレード ウィザード を使用することで、必要な手順の多くを自動化できます。

Visual J# にアップグレードするには

1. Java 言語/COM コンポーネントが JActiveX ツール (**/c2j** オプションを使用) で生成されたテンプレートを使って実装されている場合や、Java 言語コンポーネントがタイプ ライブラリ インターフェイスを実装している場合は、手順 2. の前に次の処理を行う必要があります。

   Tlbimp.exe ツール (.NET Framework および Visual Studio に付属) を使って、JActiveX テンプレートの生成に使用されたタイプ ライブラリからマネージ ラッパーを生成します。次に例を示します。

   ```
   tlbimp.exe /keyfile:interopKey.snk COMComponent.tlb
   ```

   出力されるマネージ アセンブリは、.NET Framework の sn.exe ツールで生成されたキー ペアで署名されます。これにより、Regasm の実行時に表示される "アセンブリが厳密に署名されていない" という警告を回避できます。

   ここでは、Java 言語コンポーネント用のタイプ ライブラリが存在していると仮定します。Visual J++ 6.0 では、COM DLL やコントロール プロジェクトを作成するときに、COM に公開される Java 言語コンポーネントのタイプ ライブラリが自動的に作成されます。Java 言語クラス用のタイプ ライブラリが存在しない場合は、Visual J++ 6.0 および Microsoft SDK for Java に付属の VJReg.exe ツールまたは JavaReg.exe ツールを使って作成できます。

2. Java 言語/COM コンポーネントを Visual J# コンパイラ (vjc.exe) でコンパイルします。次に例を示します。

   ```
   vjc /target:library JavaSources
   ```

   メモ：

   JActiveX ツールで (**/c2j** オプションを指定して) 生成されたテンプレートを使用して Java 言語/COM コンポーネントを実装した場合、または、Java 言語のコンポーネントがタイプ ライブラリのインターフェイスを実装している場合、Java 言語/COM コンポーネント： `vjc /r:TlbimpGeneratedWrappers JavaSources` をコンパイルするときに前の手順で生成されたマネージ ラッパー アセンブリを参照する必要があります。

   次の手順で アセンブリ登録ツール (Regasm.exe) の **/codebase** オプションを使って登録できるように、アセンブリに署名しておく必要があります。アセンブリに署名するには、System.Reflection 名前空間にある AssemblyKeyFile 属性を、プロジェクト内のファイルの 1 つに適用します。たとえば、`/** @assembly System.Reflection.AssemblyKeyFile("myKey.snk") */` のように指定します。`myKey.snk` は、.NET Framework の sn.exe ツールで生成されたキー ペア ファイルです。

   VJReg.exe ツールで生成されたタイプ ライブラリを使って COM クライアントから Java 言語/COM コンポーネントにアクセスしていた場合、COM クライアントでは、コンポーネントで公開されている **dispinterface** が使用され、メンバの DispId がキャッシュされた可能性があります。クラス メンバの DISPID は、Visual J# でも同一であるとは限りません。このため、既存の COM クライアントで問題が発生する場合があります。この問題を回避するには、Java 言語/COM コンポーネントのタイプ ライブラリが存在するのであれば、タイプ ライブラリをコンピュータに登録し、コンポーネントに対する **@com.register** ディレクティブの *typelib* パラメータでタイプ ライブラリの GUID を指定します (まだ指定されていなかった場合)。問題が発生すると、Visual J# コンパイラ (vjc.exe) は警告 VJS1553 を表示します。詳細については、「コンパイラの警告 (レベル 2) VJS1553」を参照してください。

3. Regasm.exe ツールを使って、生成された DLL または EXE を登録します。このツールは、.NET Framework および Visual Studio に付属しています。次に例を示します。

   ```
   Regasm /codebase generated_DLL_or_EXE
   ```

Visual J++ 6.0 の Java 言語/COM アプリケーションを Visual J# にアップグレードするときに、JActiveX で生成されたラッパーを手動で編集して .NET Framework COM 相互運用機能の属性や API を追加すると、予期しない問題が実行時に発生する可能性があります。Visual J++ 6.0 の **@com** や **@dll** などの属性と .NET Framework COM 相互運用機能の属性を混在させることはサポートされていません。Visual J++ 6.0 の Java 言語/COM アプリケーションをアップグレードする場合でも、Visual J# で新しいアプリケーションを記述する場合でも、これらの属性を混在させないようにすることを強く推奨します。

参照

概念

デザインの概要 : COM コンポーネントからのアクセス

その他の技術情報

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

デザインの概要：COM コンポーネントからのアクセス

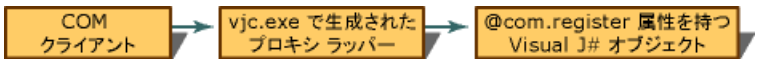
Visual J++ 6.0 アプリケーションをアップグレードするには、Visual J# コンパイラ (vjc.exe) を使ってアプリケーションをコンパイルします。vjc.exe は、Visual J++ 6.0 で COM に公開された Java 言語クラスと同じパブリック メソッド、プロパティ、およびフィールドを持つ同等のラッパー クラスをコンパイル時に作成します。ただし、ラッパー クラスのメソッドは、パラメータや戻り値として同等の .NET Framework 型を設定する点が異なります。ラッパーは、`ComVisible(true)` としてマークされます。出力された DLL に対して Regasm.exe ツールを実行すると、実際の Java 言語クラス自体が登録されるのではなく、Java 言語クラスの **@com.register** 属性で指定された CLSID を使ってラッパー クラスが登録されます。

COM クライアントは、Java 言語オブジェクトではなく、vjc.exe で生成されたラッパーと直接やり取りします。メソッドには、スクリプトを使ってアクセスしたり、**IDispatch** インターフェイスを使ってメソッドを呼び出す COM クライアントによってアクセスしたりできます。このクラスに対して生成されたラッパーとタイプ ライブラリの構造は、Visual J++ 6.0 の Java 言語オブジェクトから COM クライアントに公開されていた構造に類似しています。

ラッパーは、内部的に Java 言語オブジェクトのインスタンスを作成し、ラッパーのメソッドを Java 言語オブジェクトのメソッドに対応付け、Java 言語クラスと実際に公開されたラッパー クラスとの間のすべての違いを処理します (型のマーシャリングの実行など)。

この方法の利点は、Java 言語コンポーネントが MSJVM (Microsoft Java Virtual Machine) ではなく共通言語ランタイムで実行されていることが、COM アプリケーションで認識されないことです。COM クライアントの視点では、Java 言語オブジェクトが変更されていないように見えます。

Visual J# にアップグレードした後の Java 言語/COM コンポーネント



COM 型は適切な Java 言語型にマーシャリングされ、Java 言語のメソッドの戻り値は COM 型にマーシャリングされます。2 つの型間の対応付けは、Visual J++ 6.0 での対応付けに似ています。この対応付けには、**IUnknown** や **IDispatch** などのインターフェイス、VARIANT、SAFEARRAY、構造体、列挙型などのデータ型、およびユーザー定義のクラスとインターフェイスが含まれています。

Java 言語の例外がスローされた場合は、適切な HRESULT が COM クライアントに返されます。例外と HRESULT との間の対応付けは、Visual J++ の場合と同じです。

参照

処理手順

方法 : COM に公開される Visual J++ 6.0 コンポーネントのアップグレード

Visual J# でサポートされていないシナリオ

このセクションでは、次に示す状況でサポートされていないシナリオについて説明します。

- COM を呼び出す Java 言語
- Java 言語を呼び出す COM

参照

関連項目

サポートされていない .NET Framework 機能
サポートされていないクラス ライブラリと機能
既知の問題と問題回避

その他の技術情報

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

COM を呼び出す Java 言語

このセクションでは、次に示す状況でサポートされていないシナリオについて説明します。

- JActiveX オプションのサポート
- @com ディレクティブの部分的なサポート
- スレッド モデルの問題
- Java 言語での ActiveX コントロールのホスト

参照

関連項目

Visual J# でサポートされていないシナリオ

JActiveX オプションのサポート

Visual J++ 6.0 の COM ラッパーの追加ウィザードで作成されるラッパーは、次に示す JActiveX のオプションを使って生成されます。

```
/wfc /w /xi /X:rkc /l FileName.tmp /nologo /d ProjectName COMDLLNAME.dll
```

これらのラッパーは、Visual J# コンパイラ (vjc.exe) でコンパイルできます。次の表は、サポートされていない JActiveX オプションの一覧です。

変換

JActiveX オプション	説明	結果
/bx+	ActiveX コントロールを Java Bean として公開します。	コンパイル エラー
/X:m-	自動マーシャリングを無効にします。	無視
/X:s2	符号なし 2 バイト整数を "char" として表します。	無視
/X:vi	void ポインタを "int" として表します。	無視

出力

JActiveX オプション	説明	結果
/ci:as	すべてのソース インターフェイスの **EventInfo** エントリを作成します (WFC のみ)。	コンパイル エラー

その他

JActiveX オプション	説明	結果
/t *jnffile*	カスタム設定用の JNF ファイル (Java タイプ ライブラリ情報ファイル) を指定します (既定値は none)。	コンパイル エラー

Javatlb

JActiveX オプション	説明	結果
/x2	すべての 2 バイト整数を "char" として表します。	無視
/xh	S_FALSE を **ComSuccessException** に対応付けません。	無視
/n *jnffile*	カスタム マーシャリング用の JNF ファイルを指定します (既定値は none)。	コンパイル エラー
/G3.1 /G4	Microsoft Virtual Machine for Java Version 3.1 を対象にします (既定値は /G4)。 Microsoft Virtual Machine for Java Version 4.0 を対象にします (既定値は /G4)。	無視

表の「結果」列の値は、次のように定義されます。

コンパイル エラー

生成されたラッパーは、vjc.exe でコンパイルできないことがあります。

無視

生成されたラッパーはコンパイルできますが、ラッパー内にあるオプションに固有のコードは無視されます。このため、Visual J++ 6.0 に関するマーシャリングとスレッド モデルの非互換性の問題が実行時に発生することがあります。

Visual J# コンパイラでは、JActiveX より前にリリースされた旧バージョンのツールで出力されたラッパーをコンパイルできないことがあります。

JActiveX ツールは、Visual J# には付属していません。このツールは、Visual J++ 6.0 および Microsoft SDK for Java に付属しています。JActiveX の詳細については、Visual J++ 6.0 ドキュメント (MSDN) または Microsoft SDK for Java のドキュメントを参照してください。

参照

関連項目

COM を呼び出す Java 言語

@com ディレクティブの部分的なサポート

次の表は、サポートされている **@com** ディレクティブで無視される属性を示しています。

@com ディレクティブ	無視される属性
@com.class	**DynamicCasts**
@com.interface	スレッド
@com.method	**addFlagsVtable**
@com.parameters	**customMarshalFlags**、**thread**、**type = CUSTOM | CUSTOMBYREF | CUSTOMBYVAL**、**customMarshal**、**size**、**byref**、および **array**
@com.structmap	**customMarshalFlags**、**customMarshal**、**addFlags**、**thread**
@com.struct	**safe**、**safeAddFlags**
@com.register	**typelib**、**version**、**description**

これらのディレクティブや属性の一部がラッパーに含まれていても、ラッパーのコンパイルはできますが、Visual J++ 6.0 に関する実行時の非互換性の問題が、特にマーシャリングとスレッド処理で発生することがあります。

@com.parameters ディレクティブと **@com.structmap** ディレクティブの **customMarshalFlags**、**customMarshal**、**type = CUSTOM | CUSTOMBYREF | CUSTOMBYVAL** の各パラメータがサポートされていないため、Visual J# ではカスタム マーシャリングがサポートされていません。Visual J# コンパイラでは、JActiveX で生成されたラッパーのうち、これらのパラメータを持つディレクティブが含まれるラッパーはコンパイルできません。

iid_is 情報と **size_is** 情報をタイプ ライブラリに反映させるために、COM コンポーネントの IDL ファイルで **JTLBATTR_IID_IS** マクロと **JTLBATTR_SIZE_IS** マクロが使用されている場合は、JActiveX で生成されたラッパーに **type = CUSTOM** パラメータと **customMarshal** パラメータを使うディレクティブが含まれます。したがって、これらのマクロを使うラッパーは Visual J# コンパイラでコンパイルできません。**JTLBATTR_IID_IS** および **JTLBATTR_SIZE_IS** の詳細については、Visual J++ 6.0 または Microsoft SDK for Java のドキュメントを参照してください。

参照

関連項目

COM を呼び出す Java 言語

スレッド モデルの問題

Visual J++ 6.0 の COM コンポーネントを Visual J# で実行するときに、スレッド モデルの非互換性の問題があることがわかっています。詳細については、「COM コンポーネントのスレッド モデルの処理」を参照してください。

参照

関連項目

COM を呼び出す Java 言語

Java 言語での ActiveX コントロールのホスト

/wfc オプションを使って ActiveX コントロール用に生成した JActiveX ラッパーだけがサポートされています。/wfc は、Visual J++ 6.0 のフォーム デザイナで ActiveX コントロールを WFC フォームにドラッグ アンド ドロップしたときに使用される既定のオプションです。

参照

関連項目

COM を呼び出す Java 言語

Java 言語を呼び出す COM

サポートされていない機能を次に示します。

- Java 言語コンポーネントの COM 呼び出し可能ラッパー (CCW: COM Callable Wrapper) で公開されるインターフェイスのセットには、Visual J++ 6.0 との互換性がありません。Visual J# では、次のインターフェイスは CCW によって公開されません。
    - **IProvideClassInfo2** および **IExternalConnection**
    - **IPersist**、**IPersistStreamInit**、および **IPersistStorage** (MSJVM では、クラスが **java.io.Serializable** または **java.io.Externalizable** を実装している場合に公開される)
- **com.ms.com.NoAutoMarshaling** や **com.ms.com.NoAutoScripting** などのクラスのマーカー インターフェイスは無視されます。これらのインターフェイスの機能に依存する Visual J++ アプリケーションでは、アップグレード時に適切な変更が必要になる場合があります。詳細については、「問題と回避策 : COM を呼び出す Java 言語」を参照してください。
- 現在、COM クライアントの Java 言語/COM クラスによって発生したイベントのシンクはサポートされていません。
- VT_BYREF 型の VARIANT のマーシャリングはサポートされていません。

COM に公開される Java 言語オブジェクトでは、次の @com ディレクティブは Visual J# コンパイラ (vjc.exe) によって無視されます。

- **@com.transaction**
- **@com.typeinfo**

Visual J# では、WFC コントロールまたは Java Bean を ActiveX コントロールとして公開することはサポートされていません。

また、Visual J# では、Java 言語モニカもサポートされていません。

参照

関連項目

Visual J# でサポートされていないシナリオ

既知の問題と問題回避

このセクションでは、アップグレードに関連する既知の問題の回避について説明します。

- 問題と回避策 : COM を呼び出す Java 言語
- 問題と回避策 : Java 言語を呼び出す COM
- カスタム マーシャリングを使う Java 言語/COM コンポーネントのアップグレード

参照

関連項目

Visual J# でサポートされていないシナリオ

その他の技術情報

Java 言語/COM 相互運用機能を使ったプロジェクトのアップグレード

問題と回避策：COM を呼び出す Java 言語

このトピックでは、Visual J# クライアントから COM コンポーネントを呼び出すときに発生する可能性のある問題の解決方法について説明します。

- JActiveX ラッパーのパッケージ名が、マネージ ラッパーの生成先となる名前空間の名前と同じ場合は、コンパイラ エラーが発生します。

  この問題を解決するには、Tlbimp.exe ツールでマネージ ラッパーを生成するときに、/namespace オプションを使います。Tlbimp オプションの詳細については、「アンマネージ コードとの相互運用」および「タイプ ライブラリ インポータ (Tlbimp.exe)」を参照してください。

- Java 言語アプリケーションが、名前と GUID が同じである COM インターフェイスまたはコクラスを持つ複数の COM DLL を参照するときは、vjc.exe の /jcpa オプションを使って名前の競合を明示的に解決することが必要になる場合があります。Java 言語インターフェイス (@com ディレクティブでマークされていないインターフェイス) は、**IDispatch** から派生したインターフェイスが COM 側で必要な場合にだけ、COM にマーシャリングできます。それ以外の場合、Java 言語インターフェイスのマーシャリングはできません。現時点では、JActiveX から生成された COM インターフェイスのマーシャリングだけが完全にサポートされています。

- **com.ms.com.NoAutoMarshaling** や **com.ms.com.NoAutoScripting** などの JActiveX ラッパーのマーカー インターフェイスは無視されます。これらのインターフェイスの機能に依存している Visual J++ アプリケーションは、次の理由により、アップグレード中に適切に変更することが必要な場合があります。

  クラスに **com.ms.com.NoAutoScripting** インターフェイスを実装すると、MSJVM (Microsoft Java Virtual Machine) では、クラスの COM 呼び出し可能ラッパーの **IDispatch** の実装が公開されなくなります。これは、VBScript や Visual Basic などのスクリプト エンジンからクラスのインスタンスが呼び出されることを防ぐために使用できました。Visual J# ではこのインターフェイスがサポートされていないため、このインターフェイスを実装するクラスの COM 呼び出し可能ラッパーは、アップグレード後に **IDispatch** の実装を公開するようになります。

  com.ms.com.NoAutoMarshaling インターフェイスを実装すると、MSJVM では、クラスの COM 呼び出し可能ラッパーのフリー スレッド マーシャラが公開されなくなります。これは、Java 言語/COM オブジェクトを特定のアパートメントに配置する必要のある COM ローカル サーバーの作成に使用できました。Visual J# ではこのインターフェイスがサポートされていないため、このインターフェイスを実装するクラスの COM 呼び出し可能ラッパーは、アップグレード後にフリー スレッド マーシャラを公開するようになります。

- 現在、S_FALSE などのエラーでない HRESULT は、常に S_OK として扱われ、**com.ms.com.ComSuccessException** 型の例外はスローされません。

- 配列のマーシャリングはサポートされていません。現在のリリースの Visual J# では、配列の最初の要素だけが、Java 言語クラスから COM クライアントに、またはその逆にマーシャリングされます。

- コンパイル前に開発者によって編集された JActiveX ラッパーは、vjc.exe でコンパイルできないことがあります。コンパイルに成功した場合でも、実行時に非互換性の問題が発生する可能性があります。

- ディスパッチ インターフェイスの COM プロパティのアクセサ メソッドから何も返されないと、Java 言語クライアントでプロパティの値を読み取ろうとしたときに **ComFailException** が発生します。これを回避するには、COM サーバーが常になんらかの値を返すようにします。

- Visual J# では、次のシナリオはサポートされていません。

  - COM メソッドに対するパラメータとしての多次元のセーフ配列のマーシャリング。これは実行時に失敗します。
  - STRUCT** 型のパラメータのマーシャリング。ここで、STRUCT はタイプ ライブラリに定義されている COM 構造体です。この場合、Visual J# はコンパイル エラーを生成します。

- Visual J# での Java 言語/COM 相互運用機能のサポートは、.NET Framework の COM 相互運用機能の最上位に階層化されているため、デバッグ中のスタック トレースは Visual J++ 6.0 の場合と同じでないことがあります。

参照

関連項目

既知の問題と問題回避

問題と回避策：Java 言語を呼び出す COM

このトピックでは、COM クライアントから Visual J# コンポーネントを呼び出すときに発生する可能性のある問題の解決方法について説明します。

- **@com.register** ディレクティブでマークされていない Visual J++ 6.0 の Java 言語クラス (**/clsid** オプションを指定して JavaReg.exe ツールを使って登録されたクラスなど) を登録しようとすると、そのクラスの CLSID が自動的に生成されます。これにより、既存の COM クライアントが動作しなくなる場合があります。その場合は、**@com.register** 属性をクラスに追加し、クライアントで必要な CLSID を指定する必要があります。

- Visual J++ 6.0 では、任意の汎用 Java 言語オブジェクトを、メソッドへのパラメータとして COM に渡すことができます。COM クライアントは、**IDispatch** またはオブジェクトに実装されている特定の COM インターフェイスを使って、そのオブジェクトのメソッドにアクセスできます。

  Visual J# では、この方法で汎用 Java 言語オブジェクトを COM にマーシャリングすることはできません。オブジェクトを COM から呼び出すことができるのは、オブジェクトが次のどちらかに該当する場合だけです。

  - **@com.register** 属性付きのクラスである。
  - JActiveX ツールまたは **com.ms.com** インターフェイスによって生成された COM インターフェイスを実装している。

  オブジェクトがこれらの条件を満たしていない場合でも、そのオブジェクトを COM から呼び出すことはできますが、実行時に予期しない動作が発生することがあります。

  この制限は、.NET Framework COM 相互運用セマンティクスを利用する新しいコードを記述して、.NET Framework 型だけを使う場合には適用されません。

- Visual J++ 6.0 の Java 言語/COM コンポーネントをアップグレードするとき、そのコンポーネントが JActiveX テンプレート クラス (**/c2j**) を使って構築されたのでない場合、または JActiveX ツールによって生成されたインターフェイスを実装していない場合は、アップグレードしようとしているコンポーネントのタイプ ライブラリが存在し、コンピュータに登録されていることを確認する必要があります。そうでないと、Visual J# コンパイラは警告 VJS1553 を出力します。

  これを回避するには、Visual J# へのアップグレードを開始する前に、次のどちらかを実行します。

  - Visual J++ 6.0 に付属する vjreg.exe ツールを使って、COM に公開する Java 言語コンポーネントのタイプ ライブラリを生成および登録します。この操作には、次のコマンド ラインを使います。

    ```
    vjreg /typelib SomeTypeLib.tlb JavaComponent.class
    ```

    または

    コンピュータにコンポーネントのタイプ ライブラリが存在していても、まだ登録されていない場合は、次のコマンド ラインを使います。

    ```
    vjreg /regtypelib SomeTypeLib.tlb JavaComponent.class
    ```

- 登録して COM に公開した Java 言語マネージ コンポーネントと、Visual J++ 6.0 の対応するコンポーネントを、同じコンピュータに共存させることはできません。Java 言語マネージ クラスが COM サーバーとして登録されるときには、Visual J++ 6.0 のクラスで使用される同一の GUID で登録されます。このため、コクラスのエントリは、InProcServer32 によって msjava.dll ではなく mscoree.dll として上書きされます。その結果、クラスを Visual J++ 6.0 で使用できなくなることがあります。Visual J++ 6.0 または Microsoft SDK for Java のツールを使って .class ファイルを登録しない限り、Visual J++ 6.0 と Visual J# の両方で COM の Java 言語クラスをホストすることはできません。

- 現時点では、Java 言語コンポーネントのパブリック フィールドまたはパブリック メソッドが **getXXX** と **setXXX** という名前付けパターンに従っていても、それらがプロパティとして COM クライアントに公開されることはありません。

- Java 言語クラスが COM インターフェイスを実装している場合、COM から **IDispatch** インターフェイスを使ってアクセスできるメソッドは、その COM インターフェイスに属しているメソッドに限られます。実装されている COM インターフェイスに含まれないその他のパブリック メソッドは、COM には公開されません。Java 言語クラスが COM インターフェイスを 1 つも実装していない場合は、COM から **IDispatch** を使ってすべてのパブリック メソッドにアクセスできます (つまり、スクリプトのような遅延バインディングのシナリオが該当します)。

参照

関連項目

既知の問題と問題回避

カスタム マーシャリングを使う Java 言語/COM コンポーネントのアップグレード

Visual J++ 6.0 では、Java 言語/COM 相互運用の実行中にメソッド パラメータと構造体のカスタム マーシャリングを実行できます。タイプ ライブラリで JActiveX ツールを実行するときに /t *jnffile* オプションまたは /n *jnffile* オプションを使うと、生成されるラッパーには、カスタム マーシャリングに必要な情報が追加されます。/t オプションまたは /n オプションで使用される .jnf ファイルは、特定の Java 言語型を対応するネイティブ型にマーシャリングするために必要なカスタム マーシャラ クラスを指定します。Visual J++ 6.0 のカスタム マーシャリングに関する追加情報は、Microsoft Visual J++ 6.0 または Microsoft SDK for Java のドキュメントに記載されています。

Microsoft Visual J# では、現時点ではカスタム マーシャリングがサポートされていません。カスタム マーシャラを使って Java 言語型をネイティブ型にマーシャリングする JActiveX ラッパーをコンパイルしようとすると、コンパイル エラーが発生します。

共通言語ランタイムには、COM 相互運用機能とカスタム マーシャリングのネイティブ サポートが用意されています。ユーザーは、マネージ型から対応するネイティブ表現へのカスタム マーシャリングに使うカスタム マーシャラ クラスを指定できます。共通言語ランタイムのカスタム マーシャリングの詳細については、「カスタム マーシャリング」を参照してください。

カスタム マーシャリングを使う Java 言語/COM アプリケーションをアップグレードする方法としては、JActiveX ラッパーを使うのではなく、Tlbimp.exe ツールで生成されたマネージ ラッパーを直接使うようにコンポーネントを書き換える方法が考えられます。共通言語ランタイムでは、COM 相互運用の実行中にメソッド引数をマーシャリングするカスタム マーシャリング クラスを指定できます。カスタム マーシャラを記述することにより、Java 言語で記述されたマネージ型をネイティブ表現にマーシャリングできます。

参照

関連項目

既知の問題と問題回避

クラスを動的に読み込むアプリケーションのアップグレード

このセクションでは、次の項目について説明します。

- サポートされていない CLASSPATH 変数
- コンパイル時バインディング
- 実行時バインディング : Class.forName セマンティクス

参照

その他の技術情報

クラス ライブラリのサポート

サポートされていない CLASSPATH 変数

Visual J# では、CLASSPATH 環境変数はサポートされていません。これは、次の点に影響します。

- コンパイル時と実行時に、アプリケーションにバインドされているクラスを検索して読み込む方法
- アプリケーションでリソース ファイルを検索して読み込む方法

参照

関連項目

クラスを動的に読み込むアプリケーションのアップグレード

コンパイル時バインディング

コンパイル時にクラスがアプリケーションにバインドされる場合は、クラスを格納しているアセンブリへの参照がアプリケーションのメタデータに挿入されます。実行時にそれらのクラスが参照されると、共通言語ランタイムはアセンブリ読み込みヒューリスティック機能を使って、対応するアセンブリを検索し、読み込みます。

共通言語ランタイムによるアセンブリの検索と読み込みのしくみについては、「ランタイムがアセンブリを検索する方法」を参照してください。

参照

関連項目

クラスを動的に読み込むアプリケーションのアップグレード

実行時バインディング：Class.forName セマンティクス

Class.forName API を使って読み込まれるクラスは、アプリケーションに対して実行時に動的にバインドされます。

Class.forName(String *className***)** メソッドでは、クラスの検索に CLASSPATH 環境変数は使用されません。代わりに、次の場所から順番にクラスが検索されます。

1. 呼び出し元のアセンブリ
2. 現在のアプリケーションの AppDomain に読み込まれているすべてのアセンブリ
3. アプリケーション構成ファイル (.config) の <CODEBASE> タグで指定されたアセンブリ
4. アプリケーションの ApplicationBase
5. アプリケーション構成ファイルの <PROBING> タグで指定されたディレクトリにあるすべてのマネージ DLL

いずれかのステップでクラスが見つかると、検索は終了します。構成ファイルが見つからなかったり、構成ファイルの解析中にエラーが検出された場合、ステップ 3. と 5. は省略されます。どのステップでもクラスが見つからなかった場合は、**ClassNotFoundException** がスローされます。

Visual J# の **java.lang.Class** クラスには、アセンブリからクラスを検索して読み込むための追加のメソッドも用意されています。このメソッドの構文を次に示します。

```
Class.forName(String assemblyName, String className, boolean absolutePath)
```

このメソッドは、*assemblyName* パラメータで指定されたアセンブリから、*className* パラメータで指定されたクラスを検索します。*assemblyName* パラメータでは、アセンブリの表示名を指定する必要があります。詳細については、「AssemblyName クラス」を参照してください。ブール型のパラメータ *absolutePath* が True の場合、*assemblyName* はアセンブリの場所を指定する絶対パスを表します。*absolutePath* が False の場合は、パスが指定されていないと見なされ、共通言語ランタイムのアセンブリ検索ヒューリスティックを使ったアセンブリの検索と読み込みが行われます。指定したアセンブリにクラスが存在しない場合、または指定したアセンブリが見つからない場合は、**ClassNotFoundException** がスローされます。

詳細については、「グローバル アセンブリ キャッシュ」、「アセンブリ」、「アプリケーション ドメイン」、および「ランタイムがアセンブリを検索する方法」を参照してください。

参照

関連項目

クラスを動的に読み込むアプリケーションのアップグレード

カスタム クラス ローダーを使うアプリケーションのアップグレード

バイトコードを **Class** オブジェクトに変換するための **ClassLoader** は、サポートされていません。

サポートされていないメソッドを次に示します。

- **ClassLoader.defineClass**
- **ClassLoader.resolveClass**

Visual J# では、これらのメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

カスタム クラス ローダーは、**java.lang.ClassLoader** を拡張し、**loadClass** メソッドをオーバーライドして固有の処理を実装します。**loadClass** の実装で上の 2 つの **ClassLoader** メソッドの一方または両方が呼び出される場合は、マネージ アセンブリを読み込み、指定されたクラスをそのアセンブリから検索するように実装を変更する必要があります。

ClassLoader.defineClass と **ClassLoader.resolveClass** がサポートされなくなったため、カスタム クラス ローダーの使用時でも、**Class.getClassLoader** からは常に null が返されます。

Visual J# では、カスタム クラス ローダーを使うことは推奨されません。その代わりに、.NET Framework API を使って Microsoft Intermediate Language (MSIL) アセンブリからクラスを検索して読み込みます。

参照
関連項目
クラスを動的に読み込むアプリケーションのアップグレード

Visual J++ 6.0 のセキュリティのセマンティクスを使うアプリケーションのアップグレード

Visual J# で作成されたアプリケーションは、.NET Framework と共通言語ランタイムのセキュリティのセマンティクスに従います。これは、Visual J# で記述され、.NET Framework クラス ライブラリを利用する新しいアプリケーションと、Visual J# を使ってアップグレードされた Visual J++ 6.0 アプリケーションの両方に適用されます。

Visual J# アプリケーションでのカスタム セキュリティ マネージャ

Visual J# では、カスタム セキュリティ マネージャはサポートされていません。したがって、次に示す **java.lang.System** クラスと **java.lang.SecurityManager** クラスのメソッドはサポートされず、**com.ms.vjsharp.MethodNotSupportedException** 例外をスローします。

- **System.setSecurityManager**
- **SecurityManager.classDepth**
- **SecurityManager.classLoaderDepth**
- **SecurityManager.currentClassLoader**
- **SecurityManager.currentLoadedClass**
- **SecurityManager.getClassContext**
- **SecurityManager.inClass**
- **SecurityManager.inClassLoader**
- **System.getSecurityManager** メソッドは、常に null を返します。

既存の Visual J++ 6.0 アプリケーションをアップグレードする場合、そのアプリケーションがカスタム セキュリティ マネージャを実装していて、セキュリティ ポリシーの実装のためにこれらのメソッドを使っているときは、.NET Framework に用意されているセキュリティのセマンティクスとクラスを使うようにアプリケーションを変更する必要があります。

通常、この変更とは、**System.Security** 名前空間にある同等の API を使い、コンピュータの .NET Framework セキュリティ ポリシーを変更して必要な制限を適用することを意味します。

JDK レベル 1.1.4、WFC、およびその他の com.ms.* パッケージのクラスへのアクセス

Visual J# では、JDK レベル 1.1.4、WFC、および com.ms.* パッケージのクラスにアクセスできるのは、完全に信頼されたアプリケーションだけです。つまり、My Computer コード グループで実行されるアプリケーション、または .NET Framework セキュリティ ポリシーによって FullTrust アクセス許可セットが与えられたコード グループで実行されるアプリケーションだけが、これらのクラスにアクセスできます。部分的に信頼された呼び出し元からこれらのクラスへのアクセスを試行すると、セキュリティ例外が発生します。

ただし、Visual J# がチェックするのは、これらのクラスの直接の呼び出し元だけです。このため、開発者は、完全に信頼されていてセキュリティで保護されたライブラリをビルドし、部分的に信頼されたコードに公開できます。このようなライブラリの開発者は、ライブラリがセキュリティで完全に保護され、コンピュータの不正使用のために利用される危険がないことを保証する必要があります。

Visual J# の JDK レベル 1.1.4 のクラスは、重要な操作の実行前に適切な .NET Framework アクセス許可を要求するように設計されています。したがって、これらのクラスの呼び出し元が完全に信頼されている場合は、その呼び出し元で Deny や PermitOnly のセマンティクスを完全に適用できます。これらの信頼されている呼び出し元が、悪意を持つ可能性のある第 2 レベルの呼び出し元から呼び出される場合は、アクセス許可の要求により、ある程度のセキュリティ保護が有効になります (第 2 レベルの呼び出し元では、アクセス許可のない操作を実行できなくなります)。

ただし、JDK レベル 1.1.4 のクラスとは異なり、WFC および com.ms.* パッケージのクラスは、.NET Framework アクセス許可を要求しません。このため、これらのクラスの呼び出し元が完全に信頼されていても、その呼び出し元で Deny や PermitOnly のセマンティクスを実装することはできません。ファイル I/O などの特定のクラスは例外で、制限されていないアクセス許可セットの完全なスタック ウォークを実行します。

参照

関連項目

サードパーティのセキュリティ プロバイダを使うアプリケーションのアップグレード

その他の技術情報

クラス ライブラリのサポート

サードパーティのセキュリティ プロバイダを使うアプリケーションのアップグレード

Visual J# では、サードパーティのセキュリティ プロバイダを追加して、キーや署名の生成などの操作に使用できます。セキュリティ構成ファイルを使って、システムにインストールされているセキュリティ プロバイダと、アルゴリズムの実装を見つけるためのプロバイダの検索順序を指定できます。

セキュリティ構成ファイルの使用例については、SecurityProviders サンプルのトピックを参照してください。

Visual J# でこれらの設定を指定するには、%windir%\Microsoft.NET\Framework\v<%version%> ディレクトリにある構成ファイル vjsharp.config を使います。これは XML ファイルで、すべてのセキュリティ設定は <security> タグで指定されます。このファイルの標準的な書式を次に示します。

```
<vjsharpconfiguration>
   <security>
      <packageinfo>
         <description>Description of the package</description>
            <loadinfo class="packageName.className, assemblyName, Version=assemblyVersion,
Culture=assemblyCulture, PublicKeyToken=assemblyPublicKeyToken">
               <load/> [|| <noload/>]
            </loadinfo>
      </packageinfo>
      .
      .
      .
   </security>
</vjsharpconfiguration>
```

ファイル内のタグの使用法を次に示します。

<packageinfo>

インストールされているすべてのセキュリティ プロバイダを指定します。

<description>

パッケージの説明を入力できます。プロバイダの読み込みには影響しません。

<loadinfo>

セキュリティ プロバイダ パッケージのプロバイダ クラス名と、そのクラスが含まれているアセンブリを指定します。アセンブリがグローバル アセンブリ キャッシュにインストールされている場合、アセンブリの名前には、アセンブリのバージョン、カルチャ、および公開キー トークンも含める必要があります。アセンブリ名だけを指定した場合は、ApplicationBase を使ったアセンブリの検索だけが行われます。アセンブリがグローバル アセンブリ キャッシュにインストールされていない場合は、アセンブリの絶対パスを指定することもできます。この場合は、指定したディレクトリに指定したアセンブリが読み込まれます。

<load>または <noload>

現在の検索でアセンブリを読み込むか、または無視するかを指定します。<noload> を指定すると、アセンブリは無視されます。

既定では、インストール時に vjsharp.config ファイルが作成されることはありません。構成ファイルが存在しないときは、既定のセキュリティ プロバイダが使用されます。

別のプロバイダを使うには、この構成ファイルを %windir%\Microsoft.NET\Framework\v<Version> ディレクトリに作成します。ファイルの書式は、上で説明したとおりにする必要があります。プロバイダは、<security> タグ内に出現する順序で読み込まれます。<packageinfo> タグで書式エラーが見つかった場合は、暗黙的に無視されます。構成ファイルに指定するプロバイダのアセンブリは、アセンブリの検索と読み込みを行う .NET Framework 共通言語ランタイムのアセンブリ検索ヒューリスティックで見つけることのできる場所にインストールする必要があります。vjsharp.config ファイルを指定すると、ファイル内で指定していない限り、既定のプロバイダは検索されなくなります。既定のプロバイダの <loadinfo> タグは、次のようになります。

```
<loadinfo class="com.ms.vjsharp.security.provider.ms, vjslib, Version=1.0.4030.0, Culture=n
eutral, PublicKeyToken=B80725BFD0434260">
            <load/>
</loadinfo>
```

参照

処理手順

SecurityProviders サンプル (サードパーティのセキュリティ プロバイダの指定)

関連項目

Visual J++ 6.0 のセキュリティのセマンティクスを使うアプリケーションのアップグレード

概念

ランタイムがアセンブリを検索する方法

com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード

Visual J# では、Visual J++ 6.0 アプリケーションを Visual J# に正しくアップグレードするために必要となる **com.ms.*** パッケージがサポートされています。パッケージの一覧については、「サポートされているクラス ライブラリ」を参照してください。

.NET Framework クラス ライブラリには、サポートされていない **com.ms.*** パッケージの多くと同等の機能が用意されています。.NET Framework クラス ライブラリには、これらの **com.ms.*** クラスの多くに直接対応するクラスがあります。

次の表は、よく使用される一部の **com.ms.*** パッケージについて、それぞれと同等の .NET Framework クラス ライブラリを示しています。

com.ms.* パッケージ	.NET Framework クラス ライブラリ
com.ms.mtx	**System.EnterpriseServices**
com.ms.asp	**System.Web**
com.ms.iis.asp	**System.Web**
com.ms.util.xml	**System.Xml**
com.ms.xml.om	**System.Xml**
com.ms.xml.parser	**System.Xml**
com.ms.xml.util	**System.Xml**
com.ms.security	**System.Security** および **System.Security.Policy**
com.ms.security.permissions	**System.Security.Permissions**
com.ms.vm	**System.WeakReference**

ここでは、上の表に示した **com.ms.*** パッケージの多くから、対応する .NET Framework 名前空間に移行するためのアップグレード方法について説明します。

このセクションの内容

com.ms.xml パッケージを使うコンポーネントのアップグレード

System.Xml のクラスを使って、単純な XML ファイルを読み取るアプリケーションをアップグレードする方法を示します。

com.ms.mtx パッケージを使うコンポーネントのアップグレード

com.ms.mtx パッケージを使用したアプリケーションを、**System.EnterpriseServices** のクラスを使ってアップグレードする方法を示します。

com.ms.wfc.html パッケージを使うコンポーネントのアップグレード

com.ms.wfc.html パッケージ コードの <OBJECT> タグをアップグレードする方法を示します。

関連項目

Visual J++ 6.0 からのアップグレード

Visual J++ 6.0 プロジェクトを Visual J# にアップグレードする場合の推奨事項と手順について説明します。

com.ms.xml パッケージを使うコンポーネントのアップグレード

com.ms.xml.* パッケージのクラスは、XML ドキュメントの読み込み、読み取り、変更、および保存のために使用できます。これらのクラスについては、Visual J++ 6.0 および Microsoft SDK for Java のドキュメントで詳しく説明されています。Visual J# では、com.ms.xml.* パッケージはサポートされていません。これらのパッケージを使う Visual J++ アプリケーションをアップグレードするときは、.NET Framework の **System.Xml** 名前空間の対応するクラスに移行することをお勧めします。

System.Xml のクラスは、com.ms.xml.* パッケージと同等の機能を提供し、W3C 標準に準拠しています。com.ms.xml.* パッケージでは、**Document**、**ElementCollection**、**ElementImpl**、**Element** の各クラスが最もよく使用されます。これらに直接対応する **System.Xml** のクラスは、**XMLDocument**、**XMLNodeList**、および **XMLNode** です。

使用例

単純な XML ファイルを読み取る Visual J++ アプリケーションを、System.Xml のクラスを使って Visual J# にアップグレードする方法の例を次に示します。

SampleFile.Xml

```
<?xml version="1.0" encoding="utf-8" ?>
<bookdetails>
    <publisher>Microsoft Press</publisher>
    <book>
          <name>Designing Component-Based Applications</name>
          <price>12.45</price>
    </book>
    <book>
          <name>Professional Visual C++ MTS Programming</name>
          <price>21.90</price>
    </book>
    <book>
          <name>Essential COM+, 2nd Edition</name>
          <price>33.10</price>
    </book>
</bookdetails>
```

Visual J++ 6.0 アプリケーション

```
import com.ms.xml.om.*;
public class XMLUpgrade
{
   public static void main(String[] args)
   {
      try
      {
         // Create an XML document object and load an XML file
         Document xmlDoc = new Document();
         xmlDoc.load("SampleFile.xml");

         // Get the root node of the document
         Element xmlElement = xmlDoc.getRoot();
         // Get a collection of child nodes under the root node
         ElementCollection xmlEleCo1 = xmlElement.getChildren();
         // Get all nodes named 'book' in the collection
         ElementCollection xmlEleCo2 = (ElementCollection)
         xmlEleCo1.item("book");
         int length = xmlEleCo2.getLength();
         for (int i=0; i < length; i++)
         {
            // Retrieve the current element (book) from the collection
            Element xmlElement1 = xmlEleCo2.getChild(i);
            // Print the non-marked up text of the first child
            // (book name)
            System.out.println(xmlElement1.getChild(1).getText());
         }
      }
      catch(Exception e)
```

```
        {
            System.out.println("An exception occurred");
        }
    }
}
```

Visual J# へのアップグレード後

```
import System.Xml.*;
public class XMLUpgrade
{
   public static void main(String [] args)
   {
      try
      {
         // Create an XML document object and load an XML file
         XmlDocument xmlDoc = new XmlDocument();
         xmlDoc.Load("SampleFile.xml ");

         // Get all nodes named 'book' in the document
         XmlNodeList xnl = xd.GetElementsByTagName("book");
         int length = xnl.get_Count();

         XmlNode xn = null;
         for (int i=0; i < length; i++)
         {
            // Retrieve the current element (book) from the collection
            xn = xnl.Item(i);

            // Print the non-marked-up text of the first child
            // (book name)
            System.Console.WriteLine(xn.get_FirstChild().get_InnerText());
         }
      }
      catch (Exception e)
      {
         System.Console.WriteLine("An exception occurred");
      }
   }
}
```

メモ :

ソース コードは手動で変更する必要があります。アップグレード ウィザードでは、アプリケーションに .NET Framework クラスを追加する処理が自動的に行われることはありません。

参照

関連項目

com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード

com.ms.mtx パッケージを使うコンポーネントのアップグレード

com.ms.mtx パッケージのクラスを使うと、Java 言語の開発者が Microsoft Transaction Server (MTS) の機能を使用できるようになります。これらのクラスについては、Visual J++ 6.0 および Microsoft SDK for Java のドキュメントで詳しく説明されています。Visual J# では、**com.ms.mtx** パッケージはサポートされていません。このパッケージを使う Visual J++ アプリケーションをアップグレードするときは、.NET Framework の **System.EnterpriseServices** 名前空間の対応するクラスに移行することをお勧めします。**System.EnterpriseServices** のクラスには、**com.ms.mtx** パッケージのクラスと同等の機能が用意されています。

Visual J++ で **com.ms.mtx** パッケージを使って MTS コンポーネントを記述する場合の標準形式を次に示します。

```
public int transactMethod ()
{
    IObjectContext context;
    boolean success = false;
    IMtxComp mtxComp = null;

    // Obtain MTS context
    context = (IObjectContext) Mtx.GetObjectContext();

    // Obtain resources. For example, database connections
    // Use resources
    // Perform one piece of work for one client
    // Invoke other MTS components to do some of the work
    mtxComp = (IMtxComp) context.CreateInstance(CMtxComp.clsid, IMtxComp.iid);

    // all went well
    success = true;
    return 0;

    finally
    {
        // release resources
        if (success)
            context.SetComplete();
        else
            context.SetAbort();
    }
}
```

一般的に使用されるクラスとして
は、**com.ms.mtx.IObjectContext**、**com.ms.mtx.IGetContextProperties**、**com.ms.mtx.Context**、**com.ms.mtx.MTx** などがあります。**System.EnterpriseServices** 名前空間の **ContextUtil** クラスが **com.ms.mtx.IObjectContext** クラスおよび **com.ms.mtx.Mtx** クラスに直接相当します。.NET Framework の **System.EnterpriseServices** 名前空間には、**com.ms.mtx** パッケージのその他のクラス、たとえば **ISharedProperty**、**ISharedPropertyGroup**、**ISharedPropertyGroupManager** などに直接対応するクラスもあります。

使用例

System.EnterpriseServices のクラスを使って、Visual J++ アプリケーションを Visual J# にアップグレードする方法の例を次に示します。

Visual J++ 6.0 アプリケーション

```
import com.ms.mtx.*;

/** @com.transaction(required) */
/** @com.register(clsid=E96931CC-DFDC-497C-B695-F9EC12F8F470) */
public class MtxServer
{
   public String transactMethod(String name)
   {
      String greeting = null;
      boolean complete = false;

      try
      {
```

```
            // This would probably be a database write in
            // real world application code
            greeting = "Hello, " + name;
            complete = true;
        }

        finally
        {
            IObjectContext ctx = MTx.GetObjectContext();
            if (complete)
            {
                ctx.SetComplete();
            }
            else
            {
                ctx.SetAbort();
            }
        }
        return greeting;
    }
}
```

Visual J# へのアップグレード後

.NET Framework の **System.EnterpriseServices** 名前空間のクラスを使って同等の機能を実現する方法を、次のコードに示します。アップグレードされたコンポーネントのパブリック インターフェイスのメソッドでは、Java 言語型ではなく .NET Framework の型を使う必要があります。

```
import System.EnterpriseServices.*;
import System.Runtime.InteropServices.*;

/** @attribute GuidAttribute("E96931CC-DFDC-497C-B695-F9EC12F8F470") */
/** @attribute TransactionAttribute(TransactionOption.Required) */
// The class has to extend System.EnterpriseServices.ServicedComponent
public class MtxServer extends ServicedComponent
{
    public String transactMethod(String name)
    {
        String greeting = null;
        boolean complete = false;

        try
        {
            // This would probably be a database write in
            // real world application code
            greeting = "Hello, " + name;
            complete = true;
        }
        finally
        {
            if (complete)
            {
                ContextUtil.SetComplete();
            }
            else
            {
                ContextUtil.SetAbort();
            }
        }
        return greeting;
    }
}
```

com.ms.mtx パッケージは、**com.ms.asp** パッケージを使ってコンテキスト オブジェクトを取得するコンポーネントで使用されることもあります。この場合は、.NET Framework の **System.Web.HttpContext** クラスを使って、**Request** オブジェクトまたは **Response** オブジェクトを取得できます。

参照

関連項目

com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード

com.ms.wfc.html パッケージを使うコンポーネントのアップグレード

Visual J# は、**com.ms.wfc.html** パッケージを完全にサポートしています。ただし、<OBJECT> タグのパラメータが変更され、マネージ アセンブリからクラスを読み込むときにバージョンが認識されるようになっています。このため、**com.ms.wfc.html** パッケージを使う Visual J++ 6.0 コンポーネントをアップグレードするときは、<OBJECT> タグを含む HTML ページを適切に変更する必要があります。

Visual J++ 6.0 で使用される <OBJECT> タグを次に示します。

```
<OBJECT CLASSID="java:com.ms.wfc.html.DhModule" height=0 width=0 ... VIEWASTEXT>
<PARAM NAME=__CODECLASS VALUE=UserClass>
<PARAM NAME=CABBASE VALUE=UserCabFile.CAB>
</OBJECT>
```

Visual J# では、<OBJECT> タグを次のように使う必要があります。

```
<OBJECT CLASSID="CLSID:CF83EA5E-0FFD-4476-9BF7-6C15F7F8BDCF" height=0 width=0 ... VIEWASTEX
T>
   <PARAM NAME=CODECLASS VALUE=UserAssembly#UserClass>
</OBJECT>
```

指定項目：

`CLASSID`

ユーザー定義クラスをホストするために Visual J# で使用されるコントロールを表します。この値は変更しないでください。

`CODECLASS`

DhDocument を拡張するユーザー定義クラスと、このクラスを含むマネージ アセンブリを表します。アセンブリ名とクラス名は、"#" トークンで区切ります。アセンブリ名には、ファイル名を拡張子と共に指定する必要があります。アセンブリは、アクセス元の HTML ページが置かれている場所と同じ場所から読み込まれます。また、アセンブリの相対パスを指定することもできます。

メモ：

ユーザー定義クラスは、コンピュータにインストールされている最新バージョンの Visual J# によって常にホストされます。

メモ：

このリリースの Visual J# では、`VJSVERSION` パラメータ タグはサポートされていません。このタグを <OBJECT> タグに含めても無視されます。

たとえば、Visual J# で使用される一般的な <OBJECT> タグは次のようになります。

```
<OBJECT CLASSID="CLSID:CF83EA5E-0FFD-4476-9BF7-6C15F7F8BDCF" height=0 width=0 ... VIEWASTEX
T>
   <PARAM NAME=CODECLASS VALUE=MyAssembly.dll#MyUserClass>
</OBJECT>
```

Visual J# は、クライアント コンピュータの .NET Framework セキュリティ ポリシーで完全に信頼されているコンポーネントだけを読み込んで実行します。クライアントのセキュリティ ポリシーによって完全に信頼されていると見なされるためには、次のいずれかの条件を満たす必要があります。

- コンポーネントは、キー ペアを使って署名されている必要があります。さらに、このキー ペアを使って署名されたすべてのコンポーネントやアプリケーションに FullTrust 名前付きアクセス許可セットが与えられるように、クライアント コンピュータの .NET Framework セキュリティ ポリシーを変更する必要があります。
- コンポーネントは、Authenticode 証明書を使って署名されている必要があります。さらに、この証明書を使って署名されたすべてのコンポーネントやアプリケーションに FullTrust 名前付きアクセス許可セットが与えられるように、クライアント コンピュータの .NET Framework セキュリティ ポリシーを変更する必要があります。
- コンポーネントをホストしている Web サイトからダウンロードされたコントロールに対し、FullTrust 名前付きアクセス許可セットが与えられるように、クライアント コンピュータの .NET Framework セキュリティ ポリシーを変更する必要があります。

クライアント コンピュータによって完全には信頼されないコンポーネントは実行できません。完全には信頼されないコンポーネントを含むページが

Internet Explorer で読み込まれると、ステータス バーにエラー メッセージが表示されます。クライアントの Internet Explorer のセキュリティ設定によっては、完全には信頼されないコンポーネントを含むページが読み込まれるたびに、ActiveX に関する警告ダイアログ ボックスが表示されることがあります。コンポーネントの読み込みを続行するには、ダイアログ ボックスの [はい] をクリックする必要があります。

参照

関連項目

com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード

Visual J++ 6.0 プロジェクトからのアップグレードに関する一般的な問題

一般に、Visual J++ 6.0 のプロジェクトを Visual J# にアップグレードする前には、そのプロジェクトを Visual J++ 6.0 環境でビルドして実行できることを確認する必要があります。Visual J++ 6.0 でプロジェクトのビルド エラーが発生した場合は、アップグレード分析が不完全になり、アップグレードの問題の一部またはすべてを検出できなくなる可能性があります。

Visual J# プロジェクトと Visual J++ 6.0 プロジェクトには、多くの違いがあります。プロジェクトのアップグレードに関連する問題の一部を次に示します。

- CLASSPATH は、ユーザーが追加の参照を検索する場所を指定できるようにする Visual J++ 6.0 の機能です。この設定は、Visual J# ではサポートされていません。参照を追加するには、CLASSPATH に含まれているファイルのライブラリを作成し、そのライブラリを Visual J# プロジェクトに参照として追加します。詳細については、「プロジェクト参照の追加と削除」を参照してください。バイトコード ファイル (.class) としてしか使用できないライブラリやアプリケーションを MSIL アセンブリに変換する方法については、Visual J# バイナリ変換ツールのトピックを参照してください。
- 出力のパッケージ化は、Visual Studio のセットアップ プロジェクトを使って行われます。パッケージ化形式 (CAB、ZIP、自己解凍型実行可能ファイルなど) の情報は、新しいプロジェクトにアップグレードされません。
- パッケージ化された実行可能ファイルに格納される配置情報 (会社名や製品バージョンなど) は、Visual J# のアセンブリ プロパティとしてアップグレードされます。アセンブリ属性とその使い方の詳細については、「アセンブリ属性の設定」を参照してください。
- ビルド前とビルド後の設定は、Visual J# ではサポートされていません。Visual J++ 6.0 プロジェクトのこれらの設定は、Visual J# にアップグレードされません。Visual J# プロジェクトでカスタム ビルド ステップを使う場合は、C++ メイクファイル プロジェクトをソリューションに追加し (「メイクファイル プロジェクトの作成」を参照)、[構成の種類] プロパティを [ユーティリティ] に変更します (詳細については、「[全般] プロパティ ページ (プロジェクト)」を参照)。詳細については、「カスタム ビルド ステップとビルド イベントについて」を参照してください。
- Visual J++ 6.0 プロジェクト ソリューションのソース管理設定は、Visual J# ソリューション ファイルにアップグレードされません。Visual J# の新しいソリューション ファイル (.sln) とプロジェクト ファイル (.vjsproj) は、ソース管理に追加する必要があります。[ソース管理] メニューの [ソース管理の変更] を使うと、プロジェクトをそのサーバーの場所に再バインドできます。
- Visual J++ 6.0 のプロジェクトでは、別のプロジェクトに定義されているクラスにアクセスできます。このような Visual J++ 6.0 プロジェクトを Visual J# 開発環境で開いても、プロジェクトはコンパイルされません。次の操作を行って、プロジェクト参照を追加する必要があります。

  1. Visual J++ 6.0 ソリューションを Visual J# で開き、メッセージが表示されたら、ソリューション内の各プロジェクトをアップグレードします。
  2. 必要に応じて、vjswfc.dll と vjswfchtml.dll への参照を追加します。詳細については、「[参照の追加] ダイアログ ボックス」を参照してください。
  3. クラスが参照されているプロジェクトの [出力の種類] プロパティが、[クラス ライブラリ] プロジェクトになっていることを確認します。プロジェクトを作成します。
  4. クラスを使うプロジェクトから、クラスを含むプロジェクトへの参照を作成します。

- 複数のプロジェクトを含む Visual J++ 6.0 ソリューションでは、最初のアップグレード時に 1 つ以上のプロジェクトがアップグレードされなかった場合に、残りのプロジェクトがビルドできないプロジェクトとしてマークされます。アップグレードされたソリューションを次に開いたときには、残りのプロジェクトをアップグレードするかどうかを確認するメッセージが表示されます。これらのプロジェクトをアップグレードしても、ソリューション内ではビルドできないプロジェクトとしてマークされたままになります。このため、ソリューションのビルド時には、これらのプロジェクトがスキップされます。

  ソリューションのビルド時にこれらのプロジェクトをビルドするには、次の操作を行います。

  1. ソリューション エクスプローラでソリューション ノードを右クリックし、[プロパティ] をクリックします。
  2. [ソリューション プロパティ ページ] ダイアログ ボックスで、[構成プロパティ] フォルダの [構成] サブフォルダを選択します。
  3. ダイアログ ボックスの一番上にある [構成] ボックスの一覧の [すべての構成] をクリックします。
  4. 各プロジェクトについて、[ビルド] 列のチェック ボックスがオンになっていることを確認します。

- Visual J# では、Visual J++ 6.0 のサポート に示す、Visual J++ 6.0 の拡張機能がサポートされています。**@conditional** 属性は、同じコンパイル ユニット内のメソッドおよび呼び出し元に対してのみ有効です。詳細については、「条件付きデバッグのサポート」を参照してください。

- 複数のプロジェクトと 1 つ以上の循環依存関係を含む Visual J++ 6.0 ソリューションは、Visual J# にアップグレードできません。たとえば、Visual J++ 6.0 ソリューションにプロジェクト A とプロジェクト B が含まれており、[ソリューション中の各プロジェクト固有のクラス パスをマージする] プロパティが設定されている場合、プロジェクト A はプロジェクト B のクラスを使用でき、プロジェクト B はプロジェクト A のクラスを使用できます。このプロパティは、既定ですべてのプロジェクトに対して設定されます。

  Visual J# では、プロジェクト A とプロジェクト B の両方がクラス ライブラリ プロジェクトである場合は、このソリューションのアップグレード時に、プロジェクト A にプロジェクト B への参照が追加され、プロジェクト B にプロジェクト A への参照が追加されます。ただし、Visual J# では循環参照は使用できません。このため、Visual J# では、両方のプロジェクトを結合して 1 つのアセンブリにする必要があります。

参照

その他の技術情報

Visual J++ 6.0 からのアップグレード

条件付きデバッグのサポート

Visual J# は、Visual J++ 6.0 の **@conditional** 属性をサポートし、条件に基づくアプリケーションのデバッグが可能です。ただし、属性付きメソッドおよびメソッドの呼び出し元が、同じコンパイル ユニットに含まれていることが必要です。呼び出した側のメソッドと呼び出された側のメソッドが異なるアセンブリに存在する場合は、.NET Framework の ConditionalAttribute クラスを代わりに使用します。

参照

その他の技術情報

言語サポート

Visual J++ 6.0 と Visual J# の同時利用

Visual J++ 6.0 ソリューションのアップグレード後には、Visual J++ 6.0 と Visual J# を並行して使用できます。元の Visual J++ 6.0 ソリューションは、Visual J++ 6.0 で開いて実行できます。同時に、アップグレードしたソリューションを Visual J# で開いて実行することもできます。これにより、両方の環境のさまざまな機能を試すことができます。

新しい Visual J# ソリューション ファイル (.sln) を Visual J++ 6.0 で開くことはできません。

アップグレードを行うと、アップグレード ウィザードによって生成されたアップグレード レポートが Visual J++ 6.0 プロジェクトに追加されます。これは、元のプロジェクトと同じディレクトリにレポートが生成されるためです。アップグレード レポートは HTML ファイルであるため、Visual J++ 6.0 プロジェクトのビルドや実行には影響しません。

ソース ファイルは元のプロジェクトとアップグレードされたプロジェクトの間で共有されるため、一方のプロジェクトでソースに対して行われた変更は、もう一方のプロジェクトにも影響します。

参照

その他の技術情報

Visual J++ 6.0 からのアップグレード

Visual J# の移行

Visual J# には、Java や J++ で作成された従来のアプリケーション、アプレット、バイトコードを変換するためのウィザードおよびコマンド ライン ツールが用意されています。変換が完了すると、Java 仮想マシン (JVM) がインストールされていない環境で、J# アプリケーションを .NET Framework で実行できるようになります。

レガシ アプリケーションを変換するためのツール

次のいずれかのツールを使用すると、VJ++ や Java で作成されたアプリケーションを、短時間で Visual J# に変換できます。

- J# アップグレード ウィザードを使用すると、既存の J++ プロジェクトを自動的に Visual J# に変換できます。詳細については、「方法 : 既存の J++ コードから Visual J# プロジェクトを作成する」を参照してください。
- Visual J# バイナリ変換ツールは、Java 言語のバイトコード (.class) ファイルを Microsoft® Intermediate Language (MSIL) に変換するコマンド ライン ツールです。詳細については、「Visual J# バイナリ変換ツール (Java 言語のバイトコードから MSIL への変換用)」を参照してください。
- HTML Applet to Object Tag Converter は、Java アプレットを Microsoft J# ブラウザ コントロールに変換するツールです。Web からダウンロードできます。詳細については、「HTML Applet to Object Tag Converter」を参照してください。

サポート範囲

Visual J# 変換ツールでは、次の種類のアプリケーションをアップグレードできます。

- Visual J++ 6.0 のコードおよびプロジェクト。
- Java アプレットのソース コード (ブラウザ アプリケーションおよびアプレット スタンドアロン アプリケーション用)。
- JDBC を使用するデータベース アプリケーション。
- .NET Framework の CodeDOM。ソース コードの構造を定義する **CodeDOM** (コード ドキュメント オブジェクト モデル) のメタデータにより、複数言語でアプリケーションのテキストを表示できます。
- クライアントとリモート (またはローカル) サーバーで構成される COM 相互運用アプリケーション。

サポート範囲外

次に示すレガシ パッケージ、ライブラリ、カスタム属性を使用したコードは手動で変換できます。

- VJ++ 6.0 の CLASSPATH 機能。
- JNI (Java Native Interface) またはリモート メソッド呼び出し。
- sun.io.*、sun.net.*、netscape* の各パッケージ。
- JDK 1.1.4。

参照

処理手順

| リソースを使う Visual J++6.0 アプリケーションのアップグレード

概念

Visual J++ ユーザーへの Visual J# の紹介

その他の技術情報

Visual J++ 6.0 からのアップグレード

Visual J# バイナリ変換ツール (Java 言語のバイトコードから MSIL への変換用)

Visual J# バイナリ変換ツール (JbImp.exe) は、Java 言語のバイトコード (.class) ファイルを Microsoft® Intermediate Language (MSIL) に変換します。このツールを使うと、バイトコード ファイルとしてしか使用できない JDK 1.1.4 レベルのほとんどのライブラリやアプリケーションを MSIL アセンブリに変換できます。変換したアプリケーションは、.NET Framework 上で実行できます。

このツールは、アプリケーションやライブラリの Java 言語ソースを利用できない場合にだけ使ってください。Java 言語ソースを利用できる場合は、代わりに Visual J# コンパイラ (vjc.exe) を使うことをお勧めします。

```
JbImp [options] <class_files> [ [options] <class files> ...]
```

パラメータ

Class_files

変換する .class ファイルの名前です。この引数には、ディレクトリとファイルの両方を指定します。ディレクトリ名とファイル名には、ワイルドカードとしてアスタリスク (*) や疑問符 (?) を使用できます。CAB ファイル、ZIP ファイル、または JAR ファイルを指定することもできます。

/delaysign

アセンブリに完全に署名するか、部分的に署名するかを指定します。アセンブリに公開キーだけを含める場合は、**/delaysign** を使います。

既定では、**/delaysign** は無効です。

/delaysign オプションは、**/keyfile** または **/keycontainer** と共に使用しなければ無効になります。

/help/?

JbImp.exe のオプションについての簡単な説明を表示します。

/k[eyfile]:file

キーファイル *file* を使って、生成されるアセンブリに署名します。*file* に指定されているキー ペアを使って、厳密な名前を持つアセンブリが作成されます。一般に、このファイルは sn.exe ユーティリティを使って生成されます。ファイル名に空白が含まれている場合は、文字列を二重引用符 (" ") で囲みます。

/keycontainer:string

アセンブリに厳密な名前を付けるために、キー ペアのキー コンテナの名前を指定します。ファイル名に空白が含まれている場合は、文字列を二重引用符 (" ") で囲みます。

/linkres[ource]:file[,identifier]

マネージ リソースへのリンクを作成します。リソースの読み込みに使う論理名の指定はオプションです。既定値はファイル名です。

/m[ain]:class

実行可能ファイルのエントリ ポイントを指定します。`class` という名前のクラスを見つけるために、.class ファイルのリストが解析されます。*class* という名前のクラスが存在し、そのクラスに `public static void main(String[] args)` というシグネチャを持つメソッドがある場合は、そのメソッドがエントリ ポイントとして設定されます。それ以外の場合は、変換が停止します。

このオプションを選択しない場合は、`public static void main(String[] args)` というシグネチャのメソッドを持つ最初のクラスがエントリ ポイントになります。このシグネチャのメソッドを持つクラスが存在しない場合に、**/target:exe** オプションを選択すると、変換が停止します。

引数に指定するクラス名は、簡易名にすることも、パッケージ名を含む完全限定名にすることもできます。

/nologo

コンバータの著作権情報が表示されないようにします。

/out:file

出力ファイル名を指定し、**file** という名前の .NET Framework のアセンブリ ファイルを作成します。

/r[eference]:files

.NET Framework の Microsoft Intermediate Language (MSIL) のアセンブリ ファイルを使って、.class ファイル内のメタデータ参照を解決します。

/recurse dir

親ディレクトリ dir のサブディレクトリで、.class ファイルを再帰的に検索します。このオプションは、class_files にディレクトリ名が含まれている場合にだけ意味を持ちます。

/res[ource]:file[,identifier]

アセンブリにマネージ リソースを埋め込みます。リソースの読み込みに使う論理名の指定はオプションです。既定値はファイル名です。

/securescoping

安全なスコープ設定のための規則に従います。これは、Java 言語パッケージのスコープを .NET Framework アセンブリのスコープに対応付けるために使用できる高度なオプションです。このオプションが指定されていない場合、Java 言語のバイトコード内のパッケージ スコープのメンバとプロテクト スコープのメンバは、出力アセンブリ内のパブリック スコープに対応付けられます。

/t[arget]:library

DLL を生成します。`main` が存在する場合でも、エントリ ポイントは設定されません。

/t[arget]:exe

実行可能 (.exe) ファイルを生成します。これは **/target** オプションの既定の設定です。入力 .class ファイルのいずれかで `main` を実装する必要があります。

/t[arget]:module

MSIL モジュールを生成します。このモジュールは、共通言語ランタイムで読み込んで実行することはできません。ただし、Visual J# やその他のコンパイラでは、このモジュールを使ってモジュール内の型にアクセスできます。

/usestubrefs

未解決の参照に対して、スタブの型、フィールド、およびメソッドを使います。

参照されているクラスが入力内に存在しない場合は、該当するクラスが CLASSPATH 環境変数を使って自動的に検索され、最初に見つかった場所がエラー情報と共に表示されます。このオプションが指定されていない場合、参照先のクラスが入力内に存在しないときは、変換が停止します。このオプションが指定されている場合は、作成されるアセンブリ内に、見つからなかった各クラスのスタブの型が出力されます。このため、変換には成功しますが、見つからなかったクラス メンバへのアクセスが実行時に行われると、例外がスローされます。

サポート範囲

Visual J# バイナリ変換ツールでは、次の操作がサポートされています。

- Java 言語コードから生成された .class ファイルの変換。Visual J# バイナリ変換ツールでは、JDK 1.1.4 レベルのクラス ライブラリの機能を使うほとんどのコードを変換できます。
- デリゲートや J/Direct® など、Microsoft Visual J++® 6.0 の拡張機能を使うほとんどの .class ファイルの変換。
- 変換する .class ファイルをサブディレクトリから再帰的に検索。
- 既存の .NET Framework アセンブリ コード (DLL または EXE) のインポートと利用。
- グローバル アセンブリ キャッシュ (GAC: Global Assembly Cache) にインストールできる、厳密な名前を持つアセンブリの作成。

サポート範囲外

Visual J# バイナリ変換ツールでは、次のコードはサポートされません。

- Java 言語/COM 相互運用テクノロジを使うコード。
- JDK 1.1.4 レベルより上のクラス ライブラリの機能を使うコード。
- RMI、RNI、JNI、アプレットを含む、JDK 1.1.4 レベルのクラス ライブラリの特定の機能を使うコード。

参照

その他の技術情報

J# の演算子

Visual J# アプリケーションでのリソースの使用

エラー メッセージなどのリソース ファイル (.resx) は、CLASSPATH への格納場所の割り当てを気にすることなく、Visual J# プロジェクトに保存できます。アプリケーションをビルドすると、リンカによって出力アセンブリにリソースが埋め込まれます。実行時に、共通言語ランタイム (CLR) が .NET Framework セマンティクスを使用して、アプリケーション パスから適切なリソースを見つけて読み込みます。

.NET Framework セマンティクスをリソースのために使う利点として、次の 3 つがあります。

- 完全に管理、保護された環境でアプリケーションを実行可能
- ローカリゼーションの組み込みサポート
- アプリケーションの配置の簡略化

既存の Visual J++ 6.0 プロジェクトをアップグレードする場合、Visual J# リソース ジェネレータ (Vjsresgen.exe) コマンド ライン ツールまたは .NET Framework リソース ファイル ジェネレータ ツール (Resgen.exe) を使用して、リソース ファイルを .resx ファイル形式に変換できます。詳細については、「方法 : リソースを使う Visual J++ 6.0 アプリケーションのアップグレード」を参照してください。

参照

処理手順

方法 : リソースを使う Visual J++ 6.0 アプリケーションのアップグレード

概念

アプリケーションのリソース

Visual J# アプリケーションの配置

Visual J# で作成したアプリケーションは、サーバー上で動的にコードを生成し、コンパイルするために、Visual J# CodeDOM および Visual J# コンパイラを必要とします。XML Web サービスや Web Forms アプリケーションを配置するサーバーには、Visual J# の再頒布可能コンポーネントが必要です。つまり、Visual J# クラス ライブラリ、Visual J# コンパイラ、および Visual J# CodeDOM がインストールされている必要があります。Visual J# の再頒布可能コンポーネントを頒布するためには、Visual J# 再頒布可能パッケージを使用します。それ以外の配置の手順は、他の ASP.NET アプリケーションと同じです。

詳細については、「Windows インストーラでの配置に関するチュートリアル」を参照してください。

メモ：

すべての配布チュートリアルが Visual J# に適用できます。ただし、Visual J# コンポーネントには、Visual J# 再頒布可能パッケージと一緒に提供されている、追加の再頒布可能コンポーネントが必要です。

このセクションの内容

Visual J# 起動条件

Visual J# の再頒布可能コンポーネントがターゲット コンピュータにインストールされていることを確認する方法について説明します。

Visual J# 再頒布可能パッケージの配布

Visual J# 再頒布可能パッケージの内容について説明します。

関連するセクション

アプリケーションとコンポーネントの配置

セットアップ実行可能ファイルの作成、ファイルのパッケージ化、および Web サイトの作成について説明します。

Visual J# 起動条件

Visual J# 起動条件では、Visual J# 再頒布可能パッケージが対象のコンピュータに存在するかどうかをアプリケーションのインストール中にチェックします。セットアップ中に、Visual J# 起動条件は、.NET Framework 起動条件の SupportedRuntimes プロパティで指定された Visual J# 再頒布可能パッケージのバージョンを検索します。.NET Framework 起動条件がない場合は、Visual J# 再頒布可能パッケージの現在のバージョンとして SupportedRuntimes 値が使用されます。

必要な Visual J# 再頒布可能パッケージのバージョンが見つからない場合、インストールは中断されます。また、Message プロパティに指定されたテキストを含む [Yes/No] ダイアログ ボックスが表示されます。ユーザーが [はい] をクリックした場合は、InstallUrl プロパティで指定した場所にリダイレクトされます。既定では、リダイレクト先は Visual J# 再頒布可能パッケージのコピーをダウンロードできる Microsoft Support Web サイトになります。多くの場合、再頒布可能ファイルの格納場所を参照するように InstallUrl プロパティを変更します。たとえば、CD-ROM でアプリケーションを頒布する場合は、CD に再頒布可能ファイルを含め、InstallUrl プロパティをそのファイル パスに変更します。InstallUrl プロパティを変更する場合は、Message プロパティも変更して、インストールされる内容やインストール元の場所を説明する必要があります。

この起動条件は通常、Visual J# 再頒布可能パッケージの依存関係が検出された場合に、配置プロジェクトに自動的に追加されます。Visual J# 再頒布可能パッケージとの明示的な依存関係がない場合でも、セットアップ プロジェクトに Visual J# 起動条件を手動で追加することが必要な場合があります。たとえば、Visual J# のサードパーティ コンポーネントが圧縮された状態でセットアップに含まれている場合、配置プロジェクトは Visual J# の依存関係を検出しません。

追加された起動条件は、削除できません。その依存関係が不要になった場合には、プロジェクトを再作成してください。

メモ：

使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

セットアップ プロジェクトに Visual J# 起動条件を手動で追加するには

1. ソリューション エクスプローラで、Visual J# 起動条件を追加するセットアップ プロジェクトを右クリックします。
2. [追加] をクリックし、[マージ モジュール] をクリックします。

   [モジュールの追加] ダイアログ ボックスが表示されます。
3. VJSharpRedist_x86.msm ファイルを選択し、[開く] をクリックします。

   Visual J# マージ モジュールと起動条件がセットアップ プロジェクトに追加されます。

参照

関連項目

SupportedRuntimes プロパティ
Message プロパティ (配置)
InstallUrl プロパティ
起動条件エディタのプロパティ

その他の技術情報

配置での起動条件の管理

Visual J# 再頒布可能パッケージの配布

Visual J# で記述されたアプリケーションやコントロールを使用するには、そのアプリケーションまたはコントロールを実行するコンピュータに Visual J# 再頒布可能パッケージがインストールされている必要があります。現在のリリースでは、このパッケージは、開発者がアプリケーションと一緒に再頒布する必要があります。

Visual J# 再頒布可能パッケージ (vjredist.exe) は、Visual J# メディアに収録されているほか、Web サイト http://www.microsoft.com/japan/msdn/vjsharp からダウンロードして入手することもできます。Visual J# 再頒布可能パッケージをインストールするには、.NET Framework 再頒布可能パッケージがインストールされている必要があります。詳細については、「.NET Framework の再頒布」を参照してください。

Visual J# 再頒布可能パッケージは、.NET Framework 再頒布可能パッケージが対応しているすべてのプラットフォームに対応しています。.NET Framework 再頒布可能パッケージのシステム要件以外に Visual J# 再頒布可能パッケージで必要なシステム要件はありません。

Visual J# 再頒布可能パッケージのセットアップ プログラムは、サポートされているどのプラットフォームにも同じファイルをインストールして、サーバーとクライアントの両方の用途をサポートします。

参照

概念

.NET Framework アプリケーションの配置シナリオ

その他の技術情報

Visual J# アプリケーションの配置

.NET Framework アプリケーションの配置

.NET Framework の再頒布

Visual J# のタスク

Visual J# を使用して、次のタスクを実行できます。

このセクションの内容

一般的なプログラミング タスク

Visual J# ライブラリ クラスの実装を独自の実装で置き換える方法について説明します。

Java からの移行

Java ライクなアプリケーションおよび旧バージョンの J+ プロジェクトを Visual J# にアップグレードする方法について説明します。

J# による COM アプリケーションの統合

Visual J# COM クライアントとサーバー コンポーネント、および他の COM サーバーとのインターフェイスの作成方法について説明します。

J# アプリケーションのコンパイル

Visual Studio の J# アプリケーションをビルドする方法について説明します。

J# によるデータへのアクセス

データの問い合わせ方法について説明します。

J# プロジェクトの管理

Visual Studio を使用して J# プロジェクトを管理する方法について説明します。

J# によるシリアル化の管理

データを一時的に格納および取得する方法について説明します。

J# へのアップグレード

旧バージョンのアプリケーションを Visual J# にアップグレードする方法について説明します。

J# による Web プログラミング

再利用できる J# Web コンポーネントの作成方法について説明します。

Visual J# による Windows GUI アプリケーションの作成

Visual J# を使用してアプリケーションのユーザー インターフェイスを作成する方法について説明します。

方法 : .properties ファイルからリソースにアクセスする

Visual J# とレガシ アプリケーションの間で .properties ファイルを共有する方法を示すコード例が用意されています。

方法 : Visual J# から DLL のメソッドを呼び出す

Visual J# ライブラリから静的メソッドおよび動的メソッドを呼び出すための手順を示します。

方法 : 既存の J++ コードから Visual J# プロジェクトを作成する

Visual J++ プロジェクトを Visual J# にアップグレードするために必要な手順を示します。

方法 : J# ブラウザ コントロールを配置する

単純なコントロールを作成し、そのコントロールを Web サイトに配置し、Web ページ上に表示する方法について説明します。

方法 : J# にインターフェイスを実装する

銀行の取引を比較する **java.lang.Comparable** インターフェイスを実装する方法を示すコードが用意されています。

方法 : コンソール アプリケーションへのユーザー入力を検証する

実行時に、ユーザーが姓名を入力したことを確認するコンソール アプリケーションを作成するための手順を示します。

方法 : package スコープのクラスを記述する

スコープを持つクラスをパッケージ化する方法を示す例が用意されています。

関連するセクション

Visual J++ 6.0 からのアップグレード

Visual J# へのアプリケーションのアップグレードに関する情報、およびトピックへのリンクを提供します。

一般的なプログラミング タスク

ここでは、J# アプリケーションを拡張する方法や他のアプリケーションとやり取りする方法について説明します。

このセクションの内容

方法 : Visual J# ライブラリのクラスの実装を独自の実装に置き換える

実装をカスタマイズすることで、クラス メソッドの一般的な実装を省略する方法について説明します。

参照

その他の技術情報

J# アプリケーションのコンパイル

Java からの移行

方法：Visual J# ライブラリのクラスの実装を独自の実装に置き換える

コンパイラは vjslib.dll への暗黙的な参照を使用するため、多数の vjslib.dll クラスの実装を独自に作成できます。コード内に実装を見つけることができない場合、コンパイラはコンパイルの最終段階で vjslib.dll への参照を解決します。

独自の実装を作成するには

1. 新しいコンソール アプリケーションを作成し、Program.jsl 内のコードを次のコードで置き換えます。

```
package MyPackage;

import java.util.*;

public class simple
{
    public static void main(String[] args)
    {
        Vector v = new Vector();
    }
}
```

2. ソリューション エクスプローラ で右クリックし、ショートカット メニューの [新しい項目の追加] をクリックします。
3. [クラス] を選択し、「**Vector.jsl**」を名前として入力します。
4. 既存のコードを次のコードで置換します。

```
package MyPackage;
public class Vector
{
    public Vector() { System.out.println("Vector"); }
}

/*
public class String
{
    public String() { System.out.println("String");
}
*/

}
```

5. F5 キーを押してアプリケーションをコンパイルおよび実行します。

   独自の **Vector** クラスからコンソール出力が行われます。

6. **String** クラス全体からコメントを外します。
7. F5 キーを押してコンパイルします。

   クラス名を解決できないため、コードのコンパイルは失敗します。java.lang.Object や java.lang.String のコア クラスの実装を置き換えることはできません。

   "/target は 'exe' ですが、適切な main メソッドがコンパイルで見つかりません。" というエラーが発生します。

堅牢性の高いプログラム

- **java.lang.Object** や **java.lang.string** のコア クラスの実装を置き換えることはできません。

参照

処理手順

方法 : Visual J# コンソール アプリケーションを作成する
方法 : Visual J# から DLL のメソッドを呼び出す
方法 : J# にインターフェイスを実装する

Java からの移行

ここでは、レガシ アプリケーションを J# に移行する方法について説明します。

このセクションの内容

方法 : Java 言語アプリケーションを Visual J# でコンパイルする

レガシ Java アプリケーションを J# アプリケーションに変換する方法について説明します。

方法 : Java 言語バイトコードを MSIL コードに変換する

実行可能な Java アプリケーションを実行可能な J# アプリケーションに変換する方法について説明します。

方法 : 列挙定数を作成する

定数ではなく列挙体を使用してデータ セットを表す方法について説明します。

方法 : クラス ローダーを記述する

他のクラスを独自の基準でインスタンス化できるクラスを記述する方法について説明します。

参照

その他の技術情報

一般的なプログラミング タスク

方法：Java 言語アプリケーションを Visual J# でコンパイルする

J# では、JDK (Java Development Kit) 1.2 アプリケーションの多くをシームレスにコンパイルできます。Abstract Window Toolkit (AWT) またはネイティブな Java 機能を使用したアプリケーションをコンパイルする場合は、手動でのアップグレードはほとんど必要ありません。

ここでは、独自のプロジェクトを作成した後、作成したプロジェクトに既存の .java ファイルを追加して J# でコンパイルする手順について説明します。

メモ：
使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

JDK アプリケーションを Visual J# でコンパイルするには

1. コンソール アプリケーションを新規作成します。

   コンソール アプリケーションには、既定の J# アプリケーション ファイルが表示されます。現在、アプリケーションは J# を使用していないため、Program.jsl は必要ありません。

2. ソリューション エクスプローラで、Program.jsl ファイルを右クリックし、[プロジェクトから削除] をポイントします。

   Program.jsl がプロジェクトから除外されます (ファイルが削除されるわけではありません)。これにより、コンパイル時に、ビルドするファイルの一覧に Program.jsl が表示されなくなります。必要な場合は、Program.jsl をプロジェクトに再追加することもできます。どのような種類のファイルでも追加できますが、Visual J# でコンパイルされるのは .java または .jsl の拡張子を持つファイルだけです。

3. プロジェクト名を右クリックし、[追加]、[既存の項目] の順にポイントして、.java ファイルをプロジェクトに追加します。

4. 複数の .java ファイルを追加するには、Ctrl キーを押しながら対象のファイルをクリックします。

5. [OK] をクリックして、ファイルを新しいプロジェクトに追加します。

   追加したファイルがソリューション エクスプローラに表示されます。

6. [ファイル]、[すべてを保存] の順にクリックしてプロジェクトを保存します。

7. プロジェクト名を右クリックし、[プロパティ] をポイントします。

   [構成] ボックスの一覧に、アクティブになっている構成が示されます。

8. [構成] ボックスの一覧で、[Debug] または [Release] をクリックします。

   デバッグ構成を使用してコンパイルする場合は、コードをステップ実行するためのシンボルが追加されます。リリース構成を使用する場合は、配布用に小規模なアセンブリが作成されます。

9. F5 キーを押して、プログラムをコンパイルおよび実行します。

   または

   [デバッグ開始] をクリックします。

   または

   [ソリューションのビルド] をクリックします。

   プロジェクトがコンパイルされ、構成設定に基づいて bin\debug ディレクトリに .exe アセンブリが配置されます。

10. 画面の一番下の [出力] ウィンドウで、コンパイラのステータスを確認します。

11. プログラムがコンパイルされていないときは、[エラー一覧] タブをクリックして詳細を表示します。

12. エラーを強調表示して F1 キーを押すと、問題の解決方法に関する詳細なヘルプが表示されます。

参照

処理手順

方法 : Visual J# コンソール アプリケーションを作成する

その他の技術情報

Visual J# コンパイラ

方法：Java 言語バイトコードを MSIL コードに変換する

多くのレガシ Java アプリケーションは Visual J# に容易に変換できます。Visual J# バイナリ変換ツール (JbImp.exe) を使用すると、Java 言語のバイトコード (.class) ファイルを Microsoft Intermediate Language (MSIL) に変換できます。このコマンド ライン ツールを使用すると、開発者はほとんどの JDK 1.1.4 レベルのライブラリとアプリケーション バイトコードを MSIL アセンブリに変換できます。また、JbImp.exe を使用すると J/Direct コードも変換できます。

メモ：
このツールは、アプリケーションやライブラリの Java 言語ソースを利用できない場合にだけ使ってください。Java 言語ソースを利用できる場合は、代わりに Visual J# コンパイラ (vjc.exe) を使うことをお勧めします。

詳細については、「Visual J# バイナリ変換ツール (Java 言語のバイトコードから MSIL への変換用)」を参照してください。

バイトコードを変換するには

1. Visual Studio 2005 コマンド プロンプトを開きます。
2. 1 つの Java クラスを EXE ファイルに変換するには、単純にクラスに名前を付けます。たとえば、次のようにします。

```
jbimp /target:exe /out:myAssembly.exe myClass.class
```

3. クラスを含む JAR、ZIP、または CAB アーカイブを変換するには、単純にアーカイブ ファイルに名前を付けます。たとえば、次のようにします。

```
jbimp /target:exe /main:myMainClass /out:myAssembly.dll c:\myClass.jar
```

参照

処理手順

方法 : Java 言語アプリケーションを Visual J# でコンパイルする

方法：列挙定数を作成する

列挙定数を使用して独自のデータ型を作成すると、コードがわかりやすくなり、信頼性が向上します。J# では、**enum** キーワードを使用して独自のデータ型を作成できます。これらのデータ型を使用して、変数への割り当てが可能なすべての値を定義する名前、またはその他のリテラル値のセットを宣言できます。

たとえば、プログラムで曜日を処理する場合、次の手順に示すように `WeekDay` という新しい型を作成できます。

列挙型を作成するには

1. 列挙型と呼ばれるコンソール アプリケーションを作成します。
2. Program.jsl の内容を次のコードで置き換えます。

```
import System.*;

public class Program

{
     enum WeekDay
     {
          Sunday (0), Monday, Tuesday,
          Wednesday, Thursday, Friday,
          Saturday
     }
```

enum データ型を使用すると、コードが読みやすくなり、無効な値や予期しない値が変数に割り当てられることを防止できます。要素の値を設定するには、かっこを使用します。最初の要素の既定値はゼロですが、デモンストレーションのために、Sunday の値を `(0)` に設定しています。

3. 最後の右中かっこ (}) の前に、次の **main** メソッドを挿入します。

```
     public static void main(String[] args)
     {
          System.DateTime currentDate = System.DateTime.get_Now();
          System.out.println(currentDate.get_DayOfWeek());
          DayOfWeek ActualDay = currentDate.get_DayOfWeek();
          if ((int)WeekDay.Friday == (int)ActualDay)
          {
               System.out.println("Today is " + ActualDay + ". Complete your timecard"
);
          }
          else
          {
               System.out.println("Today is " + ActualDay + ". Complete your work");
          }
     }
```

まず、**get_Now** メソッドが現在の日付を取得します。**println** メソッドは、**ToString()** メソッドを使用する代わりに、**DateTime** と **DayOfWeek enum** をそれぞれの対応する **String** に自動的にキャストしてコンソール ウィンドウに表示することに注意します。

enum データ型の目的は、基になる整数値を非表示にしてコードを読みやすくすることですが、キャストを使用する場合には数値を扱うこともできます。たとえば、今日が金曜日であるかどうかを決定するために、比較演算子をキャストして使用できます。

```
((int)WeekDay.Tuesday == (int)ActualDay)
```

enum に値を代入することもできます。ただし、整数値の代入とテストを開始する際に、最初のデータ型宣言の一部として定義されていない値を誤って代入することもあります。その場合は、コンパイルや例外は発生せず、**enum** データ型を使用する利点の 1 つが失われることになります。

4. F5 キーを押して、プログラムをコンパイルおよび実行します。

   このプログラムを実行すると、現在の曜日が出力されます。今日が金曜日であれば、アプリケーションはタイムカードを完成させるように通知します。それ以外の場合は、通常の業務に進むように通知します。

参照

処理手順

列挙型のサンプル (列挙型の宣言)

関連項目

ユーザー定義の列挙型 (J#)

方法：クラス ローダーを記述する

アプリケーションに、独自の基準に基づいてクラスをインスタンス化するカスタム クラス ローダーが必要な場合があります。この例では、既存の Java 言語アプリケーションのカスタム クラス ローダーを変更して、マネージ アセンブリからクラスを読み込む方法を示します。次の例では、呼び出し元がユーザーであるか、管理者であるかに基づいて、共通言語ランタイム (CLR: Common Language Runtime) がクラスを読み込むしくみを示します。

メモ：

クラス ローダーが使用される状況として想定されているのは、既存の Java 言語アプリケーションによって使用される場合だけです。新しいコードでマネージ アセンブリからクラスを検索して読み込むには、カスタム クラス ローダーではなく、.NET Framework の CLR セマンティクスと API を使うことが推奨されています。

クラス ローダーを記述する場合は、次の点に注意してください。

- **java.lang.ClassLoader** クラスの **resolveClass** メソッドおよび **defineClass** メソッドはサポートされていないため、使用しないようにします。
- 既存の Java 言語アプリケーションでは、.NET Framework **Class.forName()** のいずれかのバージョンを呼び出すように **loadClass()** メソッドを書き換え、アプリケーションのマネージ アセンブリを CLR が見つけて読み込むことができるように、アプリケーション構成 (.config) ファイルに適切なエントリを作成します。マネージ アセンブリがアプリケーションの作業ディレクトリに存在する場合、構成ファイルは不要で、クラスの検索は **Class.forName()** の検索ヒューリスティックによって行われます。

クラス ローダーを記述するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。
2. [新しいプロジェクト] ダイアログ ボックスの [コンソール アプリケーション] をクリックし、[プロジェクト名] ボックスに「**ClassLoaderSample**」と入力します。[OK] をクリックします。
3. ソリューション エクスプローラで Program.jsl をダブルクリックし、次のコードを追加します。

```
class SecurityClassLoader extends ClassLoader
{
    protected Class loadClass(String className, boolean resolve)
                            throws ClassNotFoundException
    {
        if ("SecureClass" == className)
        {
            String user = System.getProperty("user.name");
            if (null == user)
            {
                System.out.println("Class was not loaded. ");
                System.out.println(
                "user not found .name property is not set.");
                return null;
            }
            else if (user.equals("Administrator"))
            {
                System.out.println(
                "Load access granted on the class 'SecureClass.'");
                return Class.forName(className);
            }
        // If user is not an administrator.
        System.out.println(className +
                    " was not loaded. Invalid class or username.");
        return null;
        }
        else if ("RegularClass" == className)
```

```
            return Class.forName(className);
        else
            throw new ClassNotFoundException();
    }

    public Class loadClass(String className) throws
           ClassNotFoundException
    {
        return loadClass(className, true);
    }
}

public class ClassLoaderSample
{
    public static void main(String args[]) throws
           java.io.IOException
    {
        SecurityClassLoader ccl = new SecurityClassLoader();

        try
        {
            Class cl;
            Object obj = null;

            System.out.println(
                "Attempting to load an instance of RegularClass.");
            cl = ccl.loadClass("RegularClass");
            if (null == cl)
                System.out.println(
                "Failed to load an instance of RegularClass.");
            else
                obj = cl.newInstance();

            if (null != obj)
                System.out.println(
              "Successfully loaded an instance of RegularClass\n.");

            obj = null;
            System.out.println(
            "Attempting to load an instance of SecureClass.");
            cl = ccl.loadClass("SecureClass");
            if (null == cl)
                System.out.println(
                "Failed to load an instance of SecureClass.");
            else
                obj = cl.newInstance();

            if (null != obj)
                System.out.println(
                "Successfully loaded an instance of SecureClass.");
        }
        catch(Exception e) {
            if (e instanceof ClassNotFoundException)
                System.out.println(
                "Unable to find the specified class.");
            else if (e instanceof InstantiationException)
                System.out.println(
           "Unable to create an instance of the specified class.");
            else if (e instanceof IllegalAccessException)
```

```
                System.out.println(
               "Do not have proper access to the class.");
        }
    }
}
```

4. 次のコードをコピーして、`SecureClass` という新しいクラスを追加します。

```
public class SecureClass
{
    SecureClass()
    {
        System.out.println(
                  "In the constructor of class SecureClass");
    }
}
```

5. `RegularClass` という 2 つ目のクラスを追加し、内容を次のコードで置き換えます。

```
public class RegularClass
{
    RegularClass()
    {
        System.out.println(
                  "In the constructor of class RegularClass");
    }
}
```

6. F5 キーを押して、サンプルをビルドおよび実行します。

参照

概念

追加のクラス ライブラリのサポート

その他の技術情報

クラス ライブラリのサポート

J# による COM アプリケーションの統合

コンポーネント オブジェクト モデル (COM : Component Object Model) は、オブジェクト ブローカ (COM サーバー) およびオブジェクトのリンクと埋め込みアプリケーション (COM クライアント) との間の相互運用性を可能にするオープン ソフトウェア アーキテクチャです。ここでは、COM サーバーとやり取りする J# クライアントを記述したり、独自の COM コンポーネントを作成したりする方法について説明します。詳細については、「COM: Component Object Model Technologies」を参照してください。

このセクションの内容

方法 : COM オブジェクトを参照する

既存の COM オブジェクトを再利用する方法について説明します。

方法 : COM コンポーネントを J# で生成する

再利用できる COM コンポーネントを作成する方法について説明します。

参照

処理手順

方法 : VJ++ で記述された COM コンポーネントにアクセスする
方法 : ブラウザで実行されるコンポーネントを作成する

方法：COM オブジェクトを参照する

COM オブジェクト コンポーネントは、ダイナミック リンク ライブラリ (.dll) または実行可能 (.exe) ファイルに含まれる実行可能コードです。コンポーネントは、1 つ以上のオブジェクトと、コンポーネント内で特定の関数を実行する独立したコード単位を提供します。各オブジェクトは、メソッド、プログラミングされた手順、プロパティ、および動作の属性を持ちます。

この例では、.NET Framework の COM 相互運用セマンティクスを使って、Visual J# アプリケーションから COM DLL にアクセスする方法を示します。

COM オブジェクトを参照するには

1. ソリューション エクスプローラで、[参照設定] を右クリックし、[参照の追加] をクリックします。
2. [COM] タブをクリックして、登録されている J++ COM オブジェクトの一覧、または他のライブラリとコントロールの一覧を表示します。
3. コンポーネントを選択して、[OK] をクリックします。
4. クライアントの .jsl ファイルに、ライブラリを参照する import (Visual J#) ステートメントを追加します。たとえば、次のようにします。

```
import COMSERVERLib.*;
```

5. new (Visual J#) キーワードを使用してオブジェクトをインスタンス化します。たとえば、次のようにします。

```
COMServer_DClass cd = new COMServer_DClass();
```

使用例

Visual J# クライアントが C++ COM ディスパッチャおよびインターフェイス サーバーと通信できるしくみを次の例に示します。クライアントは、COM サーバーのインスタンスを取得し、各サーバーのインターフェイスにバインドします。バインドした後、クライアントは両方のサーバーとの通信を開始します。また、クライアントは COMServer_D (ディスパッチ サーバー) に対してイベントを発行できます。

```
//client.jsl
import COMSERVERLib.*;

public class Client {

    public Client() {

        // Create instances of the COM classes.
        COMServer_DClass cd = new COMServer_DClass();
        ICOMServer_D cdi = (ICOMServer_D) cd;

        COMServer_IClass ci = new COMServer_IClass();
        ICOMServer_I cii = (ICOMServer_I) ci;

        String greeting = "Ping:  COMServer_D";
        System.out.println(greeting);

        // Call a method on a Dispatch interface implemented
        //by COMServer_D CoClass.
        String reply = cdi.ReturnGreeting(greeting);
        if (null == reply)
            System.Console.WriteLine("Error: Unmanaged COM server returned a null string");
        else
            System.Console.WriteLine("  [From Visual J# .NET] " +  reply);

        // Call a method on an interface implemented by
        //COMServer_I CoClass.
        greeting = "Ping:  COMServer_I";
        System.out.println(greeting);
        reply = cii.ReturnGreeting(greeting);
        if (null == greeting)
            System.Console.WriteLine("Error: Unmanaged COM server returned a null string");
```

```
        else
            System.Console.WriteLine("  [From Visual J# .NET] " +  reply);

        //Add an event handler for the 'COMEvent' event
        //of the COMServer_D CoClass.
        _ICOMServer_DEvents_COMEventEventHandler eventHandler = new _ICOMServer_DEvents_COM
EventEventHandler(this.COMEvent);
        cd.add_COMEvent(eventHandler);

        System.Console.WriteLine("\nCausing COMServer_D to fire an event");
        cdi.FireCOMEvent();
    }

    public void COMEvent() {
        System.out.println("  An event was fired by the COM class COMServer_D");
    }

    /** @attribute System.STAThread() */
    public static void main(String args[]) throws java.io.IOException {
        new Client();

        System.out.println("\nPress the Return key to continue...");
        System.in.read();
    }
}
//Dispatch ComServer D's code.
#include "stdafx.h"
#include "COMServer.h"
#include "COMServer_D.h"

////////////////////////////////////////////////////////////////////////////
// CCOMServer_D

STDMETHODIMP CCOMServer_D::InterfaceSupportsErrorInfo(REFIID riid)
{
    static const IID* arr[] =
    {
        &IID_ICOMServer_D
    };
    for (int i=0; i < sizeof(arr) / sizeof(arr[0]); i++)
    {
        if (InlineIsEqualGUID(*arr[i],riid))
            return S_OK;
    }
    return S_FALSE;
}

STDMETHODIMP CCOMServer_D::ReturnGreeting(BSTR Name, BSTR *Greeting)
{
    if (NULL == Name || NULL == Greeting)
        return E_POINTER;

    wprintf(L"\n[Unmanaged COM Server] Received string \"%s\" which I am returning to the c
lient\n", Name);
    CComBSTR ccb;
    ccb.Attach(Name);
    ccb.Append(" Service is available");
    *Greeting = ccb.Copy();
    ccb.Detach();
    return S_OK;
}

STDMETHODIMP CCOMServer_D::FireCOMEvent()
{
    Fire_COMEvent();
    return S_OK;
}
```

```
STDMETHODIMP CCOMServer_D::UseCallBack()
{
    printf_s("  [UnManaged] Method \"UseCallBack()\" called on the STA object CCOMServer_D"
);
    return S_OK;
}


//Interface ComServer I's code.
#include "stdafx.h"
#include "COMServer.h"
#include "COMServer_I.h"

////////////////////////////////////////////////////////////////////////////
// CCOMServer_I

STDMETHODIMP CCOMServer_I::InterfaceSupportsErrorInfo(REFIID riid)
{
    static const IID* arr[] =
    {
        &IID_ICOMServer_I
    };
    for (int i=0; i < sizeof(arr) / sizeof(arr[0]); i++)
    {
        if (InlineIsEqualGUID(*arr[i],riid))
            return S_OK;
    }
    return S_FALSE;
}

STDMETHODIMP CCOMServer_I::ReturnGreeting(BSTR Name, BSTR *Greeting)
{
    if (NULL == Name || NULL == Greeting)
        return E_POINTER;

    wprintf(L"\n[Unmanaged COM Server] Received string \"%s\" which I am returning to the c
lient\n", Name);
    CComBSTR ccb;
    ccb.Attach(Name);
    ccb.Append(" Service is available");
    *Greeting = ccb.Copy();
    ccb.Detach();

    return S_OK;
}

STDMETHODIMP CCOMServer_I::FireCOMEvent2()
{
    Fire_COMEvent2();

    return S_OK;
}
```

参照

処理手順

方法 : COM コンポーネントを J# で生成する

方法：COM コンポーネントを J# で生成する

COM コンポーネントには、コード内で特殊なタスクを実行できる機能が用意されています。COM コンポーネントは、強力で実用的なアプリケーションを構築するための鍵となります。COM コンポーネントの作成方法およびクライアントがコンポーネントを使用できるしくみを次の例に示します。

COM コンポーネントを生成するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト]、[OK] の順にクリックします。
2. [新しいプロジェクト] ダイアログ ボックスの [クラス ライブラリ] をクリックし、[名前] ボックスに「**IClock**」と入力します。
3. [OK] をクリックします。
4. ソリューション エクスプローラを右クリックし、[追加] をポイントして、[コンポーネント クラス] をクリックします。
5. [プロジェクト名] ボックスに「**Clock**」と入力し、[OK] をクリックします。
6. Clock.jsl の内容を次のコードで置き換えます。

```
package IClock;

import System.ComponentModel.*;
import java.util.Date;

public class Clock extends System.ComponentModel.Component
{
    private System.ComponentModel.IContainer components;
    private long startTime;
    private long stopTime;
    private long elapsedTime;

    public Clock()
    {
        components = new Container();

        //Add Clock Client to the component container.
        components.Add(this);
        InitializeComponent();
    }
    public Clock(System.ComponentModel.IContainer container)
    {
        container.Add(this);
        InitializeComponent();
    }

    public void Start()
    {
        System.out.println("Starting the timer");
        startTime = (new Date()).getTime();
    }

    public long Stop()
    {
        System.out.println("Halting the timer");
        stopTime = (new Date()).getTime();
        elapsedTime = stopTime - startTime;
        Dispose(true);
        return elapsedTime;
    }
```

```
    #region Component Designer generated code
    protected void Dispose(boolean disposing)
    {
        //Remove Clock from the container.
        if (disposing)
        {
            if (components != null)
            {
                components.Dispose();
            }
        }
        //Dispose of the container itself.
        super.Dispose(disposing);
    }

    private void InitializeComponent()
    {
        this.components = new System.ComponentModel.Container();
    }
    #endregion
}
```

7. F5 キーを押して COM サーバーをコンパイルします。

タイプ ライブラリを生成して登録するには

1. コマンド ラインを開いて、「**tlbexp Clock.dll**」と入力します。
2. 「**regasm Clock.dll**」と入力して、アセンブリがローカルで実行されるように登録します。

   または

   「**gacutil /i Clock.dll**」と入力して、アセンブリがグローバル アセンブリ キャッシュ (GAC: Global Assembly Cache) で実行されるように登録します。

クライアント コードをビルドするには

1. Visual Studio で、[ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。
2. [新しいプロジェクト] ダイアログ ボックスで、[コンソール アプリケーション] をクリックし、[名前] ボックスに「**ClockClient**」と入力します。
3. [ソリューション名] ボックスの一覧の [ConsoleApplication1] をクリックし、「**ClockClient**」と入力します。[OK] をクリックします。
4. ソリューション エクスプローラで、[ClockClient] を右クリックし、[参照の追加] をクリックします。
5. [参照] タブをクリックし、Clock.dll を COM サーバーとして追加します。
6. Clock.jsl ファイルで、既存のコードを次のコードで置き換えます。

```
package ClockClient;

import System.*;
import IClock.*;

public class Program
{
    public static void main(String[] args)
    {
        Clock myclock = new Clock();
        myclock.Start();
        for (int idx = 0; idx < 5000; idx++)
        {
```

```
            //Generate some clock cycles.
            ;
        }
        long sysClock = myclock.Stop();
        System.out.println("Time elapsed since you added and removed as a COM componen
t " + sysClock + " milliseconds");
    }
}
```

7. F5 キーを押して、アプリケーションをコンパイルおよび実行します。
8. アプリケーションの実行時に、さまざまなブレークポイントを COM サーバーに設定して myClock がコンテナとの間で受け渡しされるようすを確認できます。

堅牢性の高いプログラム

COM コンポーネントにメソッドを追加したりメソッドを変更を加えたりするたびに、tlbexp と、regasm または GAC ユーティリティを使用して、アセンブリを再エクスポートおよび再登録する必要があります。

参照

処理手順

方法 : COM オブジェクトを参照する

J# アプリケーションのコンパイル

J# のソース コードを実行可能ファイル (.exe) またはダイナミック リンク ライブラリ (DLL) アセンブリにコンパイルできます。コンパイル後のバイナリ アセンブリは、コンパイラのスイッチ設定に基づいて bin\debug または bin\release ディレクトリに存在します。

アセンブリは、次のようなさまざまな方法でビルドできます。

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、Ctrl キーと Shift キーを押しながら B キーを押して、ソリューション エクスプローラに表示されている順序または指定したプロジェクトの依存関係に基づいて 1 つ以上のプロジェクトをコンパイルします。
- [ビルド] メニューの [<プロジェクト> のビルド] をクリックして、ソリューション エクスプローラで選択したプロジェクトだけをコンパイルします。
- [デバッグ] メニューの [デバッグ開始] をクリックするか、F5 キーを押して、デバッグ シンボルと情報でプログラムをコンパイルし、デバッグ モードでアプリケーションを起動します。
- [デバッグ] メニューの [デバッグなしで開始] をクリックするか、Ctrl キーを押しながら F5 キーを押して、変更があればコンパイルを実行し、実行可能ファイルを起動します。
- また、従来のコマンド ライン スイッチを使用してコンパイラを起動することもできます。たとえば、次のようにします。

```
vjc myProgram.jsl
```

参照

関連項目

コマンド ラインからのビルド (Visual J#)

その他の技術情報

Visual J# コンパイラ オプション一覧 (アルファベット順)

コンパイラ エラー VJS0001 から VJS9999

J# によるデータへのアクセス

Java Database Connectivity (JDBC) インターフェイスを使用すると、堅牢なローカルおよびリモートのデータベース アプリケーションを作成できます。JDBC は、Java 開発者がデータベース、スプレッドシート、およびフラット ファイルから情報にアクセスするプログラムを記述するための、プログラミング フレームワークです。JDBC は通常、データベースを制御するデータベース管理ソフトウェアにかかわらず、ユーザー プログラムを "バックグラウンドで" データベースに接続するのに使用されます。

J# には JDBC-ODBC ブリッジが付属しており、ODBC クライアントとの接続がサポートされている任意のデータベースとの接続に使用できます。J# アプリケーションがデータベースに接続するために、適したドライバがインストールされ、ODBC DSN もクライアント コンピュータで設定されます。

このセクションの内容

方法 : アプリケーションで JDBC を使用する

Level 4 JDBC ドライバを使用したデータへのアクセス方法について説明します。

参照

その他の技術情報

一般的なプログラミング タスク

方法：アプリケーションで JDBC を使用する

Java データベース接続 (JDBC) は、プログラムによるデータベースへの接続を可能にするデータベース ドライバです。この接続は、アプリケーションと実際のデータベースの間のブリッジとして使用します。

Visual J# には、ODBC の上に実装された JDBC が付属しています [JDBC-ODBC ブリッジ]。アプリケーションはこのブリッジにより、クライアント コンピュータに ODBC ドライバがインストールされている任意のデータベース サーバーと Visual J# で接続できます。

1 つ目の手順では、Northwind Access データベースでユーザー ID およびパスワードの資格情報を管理するためのデータ ソース名 (DSN) を設定します。2 つ目の手順では、DSN および JdbcOdbcDriver を使用してデータベースに接続し、データを返します。コードはドライバに `Class.forName("com.ms.jdbc.odbc.JdbcOdbcDriver")` という名前を付け、サーバーおよび認証資格情報を使用して DSN に名前を付ける接続文字列を次のように組み立てます。`String url = "jdbc:odbc:vbuetools";`

最後の例は、クエリを実行します。`while (rs.next())` ステートメントを使用することにより、結果セットを繰り返し処理してデータを取得します。データがコンソールに出力されると、`con.close()` ステートメントがレコードセットを閉じ、データベース接続を無効にします。

データ ソース名を設定するには

1. [スタート] をクリックし、[コントロール パネル] をポイントし、[管理ツール] をポイントし、[データ ソース (ODBC)] をクリックします。
2. [管理ツール] 画面で [データ ソース] を選択します。
3. [ODBC データ ソース アドミニストレータ] ページの [システム DSN] タブをクリックし、[追加] をクリックします。
4. [データソースの新規作成] ページで [Driver do Microsoft Access (*.mdb)] を選択し、[完了] をクリックします。

   これはデータベースの種類を示します。独自のアプリケーションでは、データベースに対応したドライバを選択してください。SQL Server の場合は、一覧をスクロールして、使用しているバージョンの SQL Server に対応したドライバを選択してください。
5. [ODBC Microsoft Access セットアップ] ページで、[データ ソース名] ボックスに「**JDBCdsn**」と入力し、[選択] をクリックします。
6. Northwind.mdb を選択し、[OK] をクリックします。
7. [Microsoft SQL Server 用の DSN の設定] ページの [ネットワークへのログイン ID で、Windows NT の認証メカニズムを使う] をクリックします。

   このオプションにより、現在の NT のユーザー ID およびパスワードを使用してデータにアクセスできるようになります。
8. [次へ] をクリックします。
9. 接続を確認し、[OK] をクリックします。

   ODBC Administrative Tool は、サーバー名および資格情報を JDBCdsn.dsn バイナリ ファイルに保存します。実行時に、J# コードはこの情報にアクセスしてデータベースに接続します。

データにアクセスするには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。
2. [コンソール アプリケーション] をクリックし、[プロジェクト名] ボックスに「JDBC2」と入力します。
3. Program.jsl の内容を次のコードで置き換えます。

```
import java.*;
import java.sql.*;
import java.util.*;
import com.ms.jdbc.odbc.JdbcOdbcDriver;
public class Connect{

    private java.sql.Connection  con = null;
    private ResultSet rs;

     // Constructor:
     public Connect(){}
```

```
    private void FindEmployee() throws SQLException, ClassNotFoundException
    {
        try
        {
            Class.forName("com.ms.jdbc.odbc.JdbcOdbcDriver");
            String url = "jdbc:odbc:JDBCdsn";
            con = DriverManager.getConnection(url, "", "");
            Statement stmt = con.createStatement();
            rs = stmt.executeQuery("Select EmployeeID, LastName,  Title from Employees
");
            while (rs.next())
            {
                //Column names:
                System.out.println(rs.getInt("EmployeeID") + " " + rs.getString("LastN
ame") + " " + rs.getString("Title"));
            }
            if(con != null)
                con.close();
            con = null;
        }
        catch (SQLException ex)
        {
            throw ex;
        }
        catch (ClassNotFoundException clex)
        {
            throw clex;
        }
    }

     private void closeConnection(){
          try{
               if(con != null)
                    con.close();
               con = null;
          }catch(Exception ex){
               ex.printStackTrace();
          }
     }

     public static void main(String[] args) throws Exception
       {
          Connect myDbTest = new Connect();
          try
          {
              myDbTest.FindEmployee();
          }
          catch (Exception ex)
          {
            ex.printStackTrace();
            System.out.println("Error Trace in getConnection() : " + ex.getMessage());

}
}
}
```

4. 追加したコード内で、データ ソース名が Administrative Tools で作成した名前と一致することを確認します。たとえば、次のようにします。

```
String url = "jdbc:odbc: JDBCdsn";
```

5. F5 キーを押して、コードをコンパイルおよび実行します。

   プログラムは、Northwind データベースの Employees テーブルから各従業員のシリアル番号、姓、および役職をフェッチします。

セキュリティ

操作性のために、データベース名は DSN 設定に保存できます。アプリケーションでは、資格情報をハード コードしたり、安全でない場所または安全性の低い方法で保存したりしないように、適切な注意が必要です。Web アプリケーションの場合は、ユーザー ID、パスワードなどのデータベース情報をエンコード、暗号化、またはハッシュする必要があります。

参照

その他の技術情報

アクセス修飾子 (Visual J#)

J# プロジェクトの管理

Visual J# プロジェクトは、アプリケーション コード、およびストリング テーブルやフォームなどの付属リソースの管理に役立ちます。

このセクションの内容

方法 : Visual J# 再頒布パッケージのバージョン管理を設定する

プロジェクトのプロパティを変更してアプリケーションのバージョン管理を行う方法について説明します。

参照

その他の技術情報

J# へのアップグレード

方法：Visual J# 再頒布パッケージのバージョン管理を設定する

さまざまなクラスを単一の J# アセンブリとして配布する場合、JAR (Java Archive) マニフェストを使用して内容を記述する必要はありません。代わりに、その他の情報と同様に、Visual Studio プロジェクト内からバージョンを設定できます。アプリケーションを新しくリリースするたびに、**@assembly** 属性の *AssemblyVersion* パラメータを変更するだけです。次の手順では、アプリケーションまたはライブラリのバージョン番号を変更する方法を示します。

アプリケーションのソフトウェア バージョンを変更するには

1. [ファイル] メニューの [開く] をポイントし、[プロジェクト/ソリューション] をクリックします。
2. プロジェクト名を選択し、[開く] をクリックします。
3. ソリューション エクスプローラで、AssemblyInfo.jsl をクリックします。
4. AssemblyInfo.jsl ファイルで、*@AssemblyVersion* パラメータのバージョン情報を検索して更新します。たとえば、次のようにします。

   ```
   /** @assembly AssemblyVersion("2.1.0") */
   ```

5. F5 キーを押してアプリケーションを再コンパイルします。

   プロジェクトがコンパイルされ、*AssemblyVersion* パラメータで定義した現在のバージョン情報がアプリケーションの [プロパティ] タブに表示されます。

6. Windows エクスプローラで、プロジェクトの bin\debug ディレクトリまたは bin\release ディレクトリでアセンブリを見つけます。
7. 実行可能ファイル名 (.exe) またはライブラリ名 (DLL) を右クリックし、[プロパティ] をクリックします。

   [プロパティ] ウィンドウが開き、全般情報、バージョン情報、セキュリティ情報、および概要情報が表示されます。

8. [バージョン] タブをクリックし、[アセンブリ バージョン] の横の [値] ボックスの値を確認します。

   AssemblyVersion パラメータで指定したバージョン番号が表示されます (たとえば、2.1.0)。詳細については、http://msdn.microsoft.com/library/ja/default.asp?url=/library/ja/cpref/html/frlrfsystemreflectionassemblycopyrightattributeclasscopyrighttopic.asp を参照してください。

参照

その他の技術情報

Java からの移行

J# によるシリアル化の管理

シリアル化とは、別のアプリケーションまたはインスタンスが使用できるようにオブジェクトの状態やそのメンバの一部を格納するプロセスです。Visual J# は 2 つのモードのバイナリのシリアル化をサポートしています。従来の **java.io.Serializable** インターフェイスを実装する方法と、.NET Framework の BinaryFormatter クラスを使用する方法のどちらでもシリアル化が可能です。

このセクションの内容

方法 : J# クラスをバイナリとしてシリアル化する

将来使用する目的でデータを一時的に格納する方法を説明します。

方法 : .NET Framework から、シリアル化された Java オブジェクトを逆シリアル化する

各種クラスのインスタンス間でデータを格納および取得する方法について説明します。

方法 : Java アプリケーションで .NET Framework シリアル化を有効にする

.NET Framework を使用してバイナリのシリアル化を行う方法について説明します。

方法 : Java シリアル化オブジェクト (.ser) を .NET Framework シリアル化オブジェクトに変換する

バイナリのシリアル化を行うために、シリアル化された既存のアプリケーションを .NET Framework に変換する方法について説明します。

方法 : .NET Framework を使用して J# クラスのカスタム シリアライザを記述する

カスタム シリアライザを作成して、シリアル化対象のクラス メンバ データを制御する方法について説明します。

関連するセクション

一般的なプログラミング タスク

方法：J# クラスをバイナリとしてシリアル化する

シリアル化とは、他のアプリケーションまたはインスタンスが使用できるように、オブジェクトまたはオブジェクトのメンバの一部の状態を格納するプロセスです。バイナリ シリアル化は、バイナリ表現でデータを格納する方法です。

たとえば、クリップボードを出力先としてオブジェクトをシリアル化することによって、そのオブジェクトを異なるアプリケーション間で共有できます。オブジェクトをシリアル化して、ストリーム、ディスク、メモリ、ネットワーク上などの出力先に出力できます。バイナリ シリアル化を使用して、あるコンピュータやアプリケーション ドメインから別のコンピュータやアプリケーション ドメインに、リモートでオブジェクトを値渡しすることもできます。

シリアル化できるクラスは、次の条件を満たす必要があります。

- **java.io.Serializable** インターフェイスを実装する。
- パラメータを受け取らない既定のパブリック コンストラクタを含む。
- シリアル化できるフィールドを特定するか、**serialPersistentFields** メンバを使用してシリアル化できることを明示的に宣言する。または、**transient** キーワードを使用して、シリアル化できないフィールドを示す。

バイナリ シリアル化の使用方法を次の例に示します。最初の例では、**Serializable** インターフェイスを使用した場合はシリアル化が自動的に行われることを示します。2 番目の例では、シリアル化を特定のファイルに渡す方法を示します。各例では、ローカル ディレクトリにデータを格納します。

単純なクラスをシリアル化するには

1. コンソール アプリケーションを作成し、**Serialization** という名前を付けます。
2. ソリューション エクスプローラで Program.jsl が選択されていることを確認し、コードを次のコードで置き換えます。

```
package Serialize;

public class Address implements java.io.Serializable
{
    //Default CTOR Required for Serialization.
    public Address() { }

    public String Street;
    public String City;
    public String Region;
    public String PostalCode;

    public static void main(String[] args)
    {

        Address customerAddress = new Address();
        customerAddress.Street = "1111 White Street";
        customerAddress.City = "Sturtevant";
        customerAddress.Region = "WI";
        customerAddress.PostalCode = "53177";

    }
}
```

3. [ファイル] をクリックし、[すべてを保存] をクリックしてプロジェクトを格納します。
4. F5 キーを押して、コードをコンパイルおよび実行します。
5. Serialization\obj\ResGen.cache ディレクトリにあるシリアル化のバイナリ出力を確認します。

   クラスによってシリアル化が実装されため、共通言語ランタイム (CLR: Common Language Runtime) はシリアル化されたデータを自動的に ResGen.cache ディレクトリに格納します。

特定の出力ファイルにシリアル化するには

1. 2 つ目のコンソール アプリケーションを作成し、**Serialization2** という名前を付けます。
2. ソリューション エクスプローラで Program.jsl が選択されていることを確認し、コードを次のコードで置き換えます。

```
package Serialize2;

import java.io.*;

public class Address implements Serializable
{
    //Default CTOR Required for Serialization.
    public Address() { }

    public String Street;
    public String City;
    public String Region;
    public String PostalCode;

    public void CacheAddress()
    {
        try
        {
            // Various address checks should be put here:
            if (this.Street.length() > 0)
            {
                FileOutputStream fout = new FileOutputStream("serial.bin");
                ObjectOutputStream out = new ObjectOutputStream(fout);
//This is where the serialization takes place:
                out.writeObject(this);
                out.flush();
                out.close();
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
    public Address RetrieveAddress()
    {
        Address address2 = new Address();
        try
        {
            FileInputStream fi = new FileInputStream("serial.bin");
            ObjectInputStream fs = new ObjectInputStream(fi);
     // Deserialazation:
            address2 = (Address)fs.readObject();
            fs.close();
        }
        catch (Exception ex)
        {
            System.out.println(ex.toString());
        }
        return address2;
    }
        public String toString()
    {
        return "" + Street + " " + City + " " + Region + " " + PostalCode;
    }
```

```
    public static void main(String[] args)
    {
        Address customerAddress = new Address();
        customerAddress.Street = "1111 White Street";
        customerAddress.City = "Sturtevant";
        customerAddress.Region = "WI";
        customerAddress.PostalCode = "53177";
        customerAddress.CacheAddress();

        Address cachedAddress = new Address();
        cachedAddress = customerAddress.RetrieveAddress();
        System.out.println(cachedAddress.toString());
    }
}
```

3. [ファイル]、[すべてを保存] の順にクリックしてプロジェクトを保存します。
4. F5 キーを押して、コードをコンパイルおよび実行します。

   シリアル化は、CLR が `out.writeObject(this);` ステートメントを実行しているときに発生します。

   コンソール出力の内容は "1111 White Street Sturtevant WI 53177" です。
5. Serialization2\bin\debug\serial.bin ディレクトリにあるシリアル化のバイナリ出力を確認します。

   この例では、完全パスを指定せずに **ObjectOutputStream** ステートメントを使用しているため、デバッグ ディレクトリにバイナリが存在します。

堅牢性の高いプログラム

- シリアル化するフィールドには、**static** または **transient** のアクセス レベルを設定できません。
- パフォーマンスが問題になる場合は、XML シリアル化ではなく、バイナリ シリアル化を使用します。
- 必要に応じて、特殊シリアル化の Javadoc タグである @serial と @serialData を使用して、シリアル化されたオブジェクトを記述します。詳細については、「方法 : Javadoc コメントから XML ドキュメントを生成する」を参照してください。

セキュリティ

セキュリティ上の理由から、シリアル化されたクラスに、ユーザー ID、パスワード、またはその他の機密情報を含めないでください。

参照

その他の技術情報

シリアル化の概念

バイナリ シリアル化

方法：.NET Framework から、シリアル化された Java オブジェクトを逆シリアル化する

逆シリアル化は、アプリケーションが既に永続化した格納データを取得するプロセスです。取得したデータは、共通言語ランタイム (CLR: Common Language Runtime) によって、同一のフィールドを持つインスタンス化されたオブジェクトに割り当てられます。

逆シリアル化を行う場合は、次の点に留意します。

- クラスをシリアル化するには、パラメータを受け取らない既定のコンストラクタが必要です。
- 逆シリアル化したデータは、シリアル化したデータと順序が一致するクラスにキャストする必要があります。
- シリアル化は、パブリック データに対してのみ実行できます。
- 逆シリアル化したデータのデータ型と、そのデータを受け取ったクラスのデータ型が一致しないときは、ランタイム例外が発生します。

次の例では、基本顧客情報をシリアル化および逆シリアル化することによってアプリケーション情報をキャッシュします。

データを逆シリアル化するには

1. **BinaryFormatter** というコンソール アプリケーションを作成します。
2. ソリューション エクスプローラで Program.jsl をクリックし、コードを次のコードで置き換えます。

```
package BinaryFormatter;
//Serialize, and then deserialize.
import System.IO.*;
import System.Runtime.Serialization.*;
import System.Runtime.Serialization.Formatters.Binary.*;

/** @attribute System.SerializableAttribute() */
public class Address
{
    //Default CTOR Required for serialization.
    public Address() { }

    public String Street;
    public String City;
    public String Region;
    public String PostalCode;

    public void CacheAddress()
    {
        try
        {
            // Various address checks go here:
            if (this.Street.length() > 0)
            {
                FileStream fs = new FileStream("serial.text", FileMode.OpenOrCreate);
                BinaryFormatter bf = new BinaryFormatter();
                //Serialization starts here.
                bf.Serialize(fs, this);
                fs.Flush();
                fs.Close();
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
```

```
        }
    }
    public Address RetrieveAddress()
    {
        Address address2 = new Address();
        try
        {
            FileStream fs = new FileStream("serial.text", FileMode.Open);
            BinaryFormatter bf = new BinaryFormatter();
            //Deserialization starts here.
            address2 = (Address)bf.Deserialize(fs);
            fs.Close();
        }
        catch (Exception ex)
        {
            System.out.println(ex.toString());
        }
        return address2;
    }
        public String toString()
    {
        return "" + Street + " " + City + " " + Region + " " + PostalCode;
    }

    public static void main(String[] args)
    {
        Address customerAddress = new Address();
        customerAddress.Street = "1111 White Street";
        customerAddress.City = "Sturtevant";
        customerAddress.Region = "WI";
        customerAddress.PostalCode = "53177";
        customerAddress.CacheAddress();

        Address cachedAddress = new Address();
        cachedAddress = customerAddress.RetrieveAddress();
        System.out.println(cachedAddress.toString());
    }
}
```

3. F5 キーを押して、プログラムをコンパイルおよび実行します。

   データが正常に格納され、アドレスが取得されます。出力内容は "1111 White Street Sturtevant WI 53177" です。

4. BinaryFormatter\bin\debug\ ディレクトリの serial.bin ファイルを表示します。

参照

処理手順

方法 : J# クラスをバイナリとしてシリアル化する

その他の技術情報

シリアル化の概念

バイナリ シリアル化

方法：Java アプリケーションで .NET Framework シリアル化を有効にする

.NET **BinaryFormatter** クラスとカスタム属性を使用して .NET Framework のシリアル化を制御できます。このクラスは、より安全な表示できない形式で他のクラスのデータを格納します。

J# アプリケーション データをシリアル化または格納する方法を次の例に示します。最初の例は、コード `/** @attribute System.SerializableAttribute() */` を使用してクラス全体のデータを格納する方法を示しています。

2 番目の例は、@attribute **System.AttributeUsageAttribute** を使用した 1 つ以上のクラス メンバのシリアル化を示しています。*valueon* パラメータを使用すると、構成要素、戻り値、イベントなど、その他のアプリケーション要素をシリアル化することもできます。

詳細については、「`AttributeTargets`」を参照してください。

クラス全体のデータを格納するには

1. **`BinaryFormatter`** というコンソール アプリケーションを作成します。
2. ソリューション エクスプローラで `Program.jsl` をクリックし、コードを次のコードで置き換えます。

```
package BinaryFormatter;

import System.IO.*;
import System.Runtime.Serialization.*;
import System.Runtime.Serialization.Formatters.Binary.*;

/** @attribute System.SerializableAttribute() */
public class Address
{
    //Default CTOR Required for Serialization.
    public Address() { }

    public String Street;
    public String City;
    public String Region;
    public String PostalCode;

    public void CacheAddress()
    {
        try
        {
            // Various address checks go here.
            if (this.Street.length() > 0)
            {
                FileStream fs = new FileStream("serial.text", FileMode.OpenOrCreate);
                BinaryFormatter bf = new BinaryFormatter();
                //Serialization starts here.
                bf.Serialize(fs, this);
                fs.Flush();
                fs.Close();
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
    public Address RetrieveAddress()
    {
        Address address2 = new Address();
        try
        {
            FileStream fs = new FileStream("serial.text", FileMode.Open);
            BinaryFormatter bf = new BinaryFormatter();
            //Deserialization starts here.
            address2 = (Address)bf.Deserialize(fs);
            fs.Close();
        }
        catch (Exception ex)
        {
            System.out.println(ex.toString());
```

```
        }
        return address2;
    }
        public String toString()
    {
        return "" + Street + " " + City + " " + Region + " " + PostalCode;
    }

    public static void main(String[] args)
    {
        Address customerAddress = new Address();
        customerAddress.Street = "1111 White Street";
        customerAddress.City = "Sturtevant";
        customerAddress.Region = "WI";
        customerAddress.PostalCode = "53177";
        customerAddress.CacheAddress();

        Address cachedAddress = new Address();
        cachedAddress = customerAddress.RetrieveAddress();
        System.out.println(cachedAddress.toString());
    }
}
```

3. F5 キーを押して、プログラムをコンパイルおよび実行します。

4. BinaryFormatter\bin\debug\ ディレクトリの serial.bin ファイルを表示します。

変数だけをシリアル化するには

1. **SerializeSomeParms** というコンソール アプリケーションを作成します。

2. Program.jsl の内容を次のコードで置き換えます。

```
//** @attribute System.AttributeUsageAttribute(ValidOn = Parameter, AllowMultiple = fa
lse, Inherited = true) */
package SerializeSomeParms;
//AttributeUsageAttribute(ValidOn = Parameter

import System.IO.*;
import System.Runtime.Serialization.*;
import System.Runtime.Serialization.Formatters.Binary.*;

public class Address
{
    //Default CTOR Required for Serialization.
    public Address() { }
//** @attribute System.AttributeUsageAttribute(ValidOn = Parameter, AllowMultiple = fa
lse, Inherited = true) */

    public String Street;

    public String City;
    public String Region;
    public String PostalCode;

    public void CacheAddress()
    {
        try
        {
            // Various address checks go here.
            if (this.Street.length() > 0)
            {
                FileStream fs = new FileStream("serial.bin", FileMode.OpenOrCreate);
                BinaryFormatter bf = new BinaryFormatter();
                //Serialization starts here.
                bf.Serialize(fs, this.Street);
                fs.Flush();
                fs.Close();
            }
        }
        catch (IOException ex)
        {
```

```
                System.out.println(ex.toString());
            }
        }

        public String RetrieveStreet()
        {
            String Street = new String();
            try
            {
                FileStream fs = new FileStream("serial.bin", FileMode.Open);
                BinaryFormatter bf = new BinaryFormatter();
                //Deserialize just the street; otherwise,
                //a runtime error occurs.
                Street = (String)bf.Deserialize(fs);
                fs.Close();
            }
            catch (Exception ex)
            {
                System.out.println(ex.toString());
            }
            return Street;
        }
        public static void main(String[] args)
        {
            Address customerAddress = new Address();
            customerAddress.Street = "1111 White Street";
            customerAddress.City = "Sturtevant";
            customerAddress.Region = "WI";
            customerAddress.PostalCode = "53177";
            customerAddress.CacheAddress();

            String street = (String)customerAddress.RetrieveStreet();
            System.out.println(street);
        }
    }
```

`@attribute` *`System.AttributeUsageAttribute`* は、`Street` データだけを保存するように設定されています。

3. `F5` キーを押して、プログラムをコンパイルおよび実行します。

4. コンソール アプリケーションが番地 `"1111 White Street"` を返すことを確認します。

参照

処理手順

方法 : .NET Framework から、シリアル化された Java オブジェクトを逆シリアル化する

方法：Java シリアル化オブジェクト (.ser) を .NET Framework シリアル化オブジェクトに変換する

Java のシリアル化されたアプリケーションを、.NET Framework のシリアル化されたオブジェクトを含む Visual J# アプリケーションに変換する方法を次の例に示します。

Java シリアル化から .Net Framework シリアル化に変換するには

1. **ConvertSerialization** というコンソール アプリケーションを作成します。
2. Program.jsl ファイルに、次の Java コードをコピーします。

```
package ConvertSerialization;

import java.io.*;

public class Address implements Serializable
{
    //Default CTOR Required for Serialization.
    public Address() { }

    public String Street;
    public String City;
    public String Region;
    public String PostalCode;

    public void CacheAddress()
    {
        try
        {
            // Various address checks go here.
            if (this.Street.length() > 0)
            {
                FileOutputStream fout = new FileOutputStream("serial.bin");
                ObjectOutputStream out = new ObjectOutputStream(fout);
       //Serialization starts here.
                out.writeObject(this);
                out.flush();
                out.close();
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
    public Address RetrieveAddress()
    {
        Address address2 = new Address();
        try
        {
            FileInputStream fi = new FileInputStream("serial.bin");
            ObjectInputStream fs = new ObjectInputStream(fi);
            address2 = (Address)fs.readObject();     //Deserialazation starts here.
            fs.close();
        }
        catch (Exception ex)
```

```
        {
            System.out.println(ex.toString());
        }
        return address2;
    }
        public String toString()
    {
        return "" + Street + " " + City + " " + Region + " " + PostalCode;
    }

    public static void main(String[] args)
    {
        Address customerAddress = new Address();
        customerAddress.Street = "1111 White Street";
        customerAddress.City = "Sturtevant";
        customerAddress.Region = "WI";
        customerAddress.PostalCode = "53177";
        customerAddress.CacheAddress();

    }
}
```

3. import ステートメント `import java.io.*;` を次の J# コードで置き換えます。

```
import System.IO.*;
import System.Runtime.Serialization.*;
import System.Runtime.Serialization.Formatters.Binary.*;
```

これらのパッケージは、実行時にデータをどのようにシリアル化するかを制御します。

4. **T:System.SerializableAttribute** を追加し、クラス宣言を次のように更新します。

```
/** @attribute System.SerializableAttribute() */
public class Address
```

カスタム属性である **SerializableAttribute** はクラス全体のデータを格納します。また、Java Serialization クラスを実装しないことに留意します。代わりに、BinaryFormatter クラスをインスタンス化してデータをシリアル化します。

5. `FileOutputStream fout = new FileOutputStream("serial.bin")` ステートメントを類似した **FileStream** ステートメントに置き換えます。

```
FileStream fout = new FileStream("serial.bin", FileMode.OpenOrCreate);
```

FileMode.OpenOrCreate 列挙型は、アプリケーションが serial.bin という名前の既存のファイルを開くか、このファイルが存在しない場合に新しいファイルを作成することを指定します。

6. `ObjectOutputStream out = new ObjectOutputStream(fout);` ステートメントを **BinaryFormatter** ステートメントに置き換えます。

```
BinaryFormatter out = new BinaryFormatter();
```

このコード行は、入出力ストリームでデータのシリアル化を実際に実行する **BinaryFormatter** クラスをインスタンス化します。

7. `out.writeObject(this);` ステートメントを次のコードで置き換えます。

```
//Serialization:
out.Serialize(fout, this);
```

プログラムの実行中に、このステップでシリアル化が発生します。**this** キーワードは、**CacheAddress** クラスのすべてのデータがシリアル化されることを示します。

8. **flush** メソッドと **close** メソッドの名前の先頭は、次のように大文字で表記します。

```
fout.Flush();
fout.Close();
```

9. ［ファイル］ メニューの ［すべてを保存］ をクリックして、プロジェクトを保存します。

10. 変換されたプログラムをコンパイルして実行します。

    BinaryFormatter クラスは ConvertSerialization\bin\debug\serial.bin ファイルを格納します。

.NET Framework 逆シリアル化に変換するには

1. ConvertSerialization プロジェクトが開いていない場合は、開きます。

2. Address() コンストラクタの末尾に、逆シリアル化コードを含む Java **RetrieveAddress** メソッドを追加します。

```
    public Address RetrieveAddress()
    {
        Address address2 = new Address();
        try
        {

            FileInputStream fi = new FileInputStream("serial.bin");
            ObjectInputStream fs = new ObjectInputStream(fi);
            address2 = (Address)fs.readObject();     //Deserialazation starts here.
            fs.close();

        }
        catch (Exception ex)
        {
            System.out.println(ex.toString());
        }
        return address2;
    }

        public String toString()
    {
        return "" + Street + " " + City + " " + Region + " " + PostalCode;
    }
```

3. ストリーム クラスを次のコードで置き換えます。

```
FileStream fs = new FileStream("serial.bin", FileMode.Open);
BinaryFormatter bf = new BinaryFormatter();
//Deserialize here.
address2 = (Address)bf.Deserialize(fs);
fs.Close();
```

FileStream のインスタンスが seral.bin から格納されたデータを開こうとします。失敗した場合は、 Exception **catch** ブロックがエラーを処理します。また、データを各フィールドに正しくマーシャリングするため、**Deseserialize** メソッドは **Address** クラスへのクラス キャストを要求します。

4. 出力を確認するには、次のコードを **main** に追加します。

```
Address cachedAddress = new Address();
cachedAddress = customerAddress.RetrieveAddress();
System.out.println(cachedAddress.toString());
```

5. F5 キーを押して、コードをコンパイルおよび実行します。

   コンソール出力に、"1111 White Street Sturtevant WI 53177" という住所が表示されます。

参照

処理手順

方法 : J# クラスをバイナリとしてシリアル化する
方法 : Java アプリケーションで .NET Framework シリアル化を有効にする

方法：.NET Framework を使用して J# クラスのカスタム シリアライザを記述する

カスタム シリアライザを使用すると、シリアル化が必要なクラス メンバ データを制御できます。これは、精密なデータを格納するアプリケーションでは特に便利です。このクラスには、ネイティブ データ型を格納するさまざまなメソッドが用意されています。

Externalizable クラスを使用して GPS 座標を格納および取得する方法を次の例に示します。**readDouble** メソッドおよび **writeDouble** メソッドによって、マップ座標を精密に指定できます。

カスタム シリアライザを記述するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。
2. [コンソール アプリケーション] をクリックし、[名前] ボックスに「**CustomSerial**」と入力します。
3. ソリューション エクスプローラでで Program.jsl をクリックし、コードを次のコードで置き換えます。

```
package CustomSerial;
import java.io.*;

class CustomSerial implements Externalizable
{
    double XCoordinate;
    double YCoordinate;

    public CustomSerial() { }

    public CustomSerial(double Latitude, double Longitude)
    {
        this.XCoordinate = Latitude;
        this.YCoordinate = Longitude;
    }

    //Stores the data.
    public void writeExternal(ObjectOutput out) throws IOException
    {
        out.writeDouble(XCoordinate);
        out.writeDouble(YCoordinate);

    }

    //Reconstructs the data to an object.
    public void readExternal(ObjectInput in) throws IOException
    {
        XCoordinate = in.readDouble();
        YCoordinate = in.readDouble();
    }

    public String toString()
    {
        return "A{" +
                "x=" + XCoordinate + ", y=" + YCoordinate +
                "}";
    }
    public static void main(String[] args)
    {
        try
        {
```

```
            FileOutputStream fout = new FileOutputStream("serial.bin");
            ObjectOutputStream out = new ObjectOutputStream(fout);
            CustomSerial GPSCoordinates = new CustomSerial(-42.064478, 19.2179124);

            //Serialize with custom serializer.
            GPSCoordinates.writeExternal(out);
            out.flush();
            out.close();

            FileInputStream fin = new FileInputStream("serial.bin");
            ObjectInputStream in = new ObjectInputStream(fin);

            //Deserialize here.
            GPSCoordinates.readExternal(in);
            System.out.println("Retrieving Stored coordinates as " + GPSCoordinates.to
String());
            in.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

4. F5 キーを押してアプリケーションをコンパイルおよび実行します。

5. コンソール出力で、両方の入力値をプログラムの入力値と比較することにより、デシリアライザが両方の値を正確に取得したことを確認します。出力内容は "Retrieving Stored coordinates as 42.064478 19.2179124" です。

   シリアライザをカスタマイズすると、軽度なデータ エラーが発生する場合があります。前の例では、シリアライザで **double**、**int** の順に格納する場合、最初に **double** を読み取り、次に **int** を読み取る必要があります。そうしないと、データが正しく表示されない場合があります。同様に、前の例の値を反転させた場合、座標値はマップ上のまったく異なる場所を指すことになります。

   サブクラス化されたデータをシリアル化する場合は、外部化可能にすることも必要です。そうしないと、サブクラス化されたデータはシリアル化されません。

参照

その他の技術情報

シリアル化の概念
バイナリ シリアル化

J# へのアップグレード

ここでは、Visual J# を使用して J++ アプリケーションをアップグレードおよび統合する方法について説明します。

このセクションの内容

方法 : Visual J++ プロジェクトを変換する

Visual J# Professional または Express でプロジェクトを開く方法について説明します。

方法 : J++/ActiveX アプリケーションをアップグレードする

J++ ActiveX アプリケーションを Visual J# に変換する方法について説明します。

方法 : VJ++ で記述された COM コンポーネントにアクセスする

新しい Visual J# アプリケーションで J++ コンポーネントを再利用する方法について説明します。

参照

概念

Visual J# の移行

方法：Visual J++ プロジェクトを変換する

Visual J# には、Visual J# アップグレード ウィザードおよび変換ウィザードが付属します。これらのウィザードは、Visual J++ 6.0 から Visual J# へのプロジェクトの変換を支援します。独自の Visual C++ プロジェクトを開いて変換する方法を次の例に示します。まず、J# アップグレード ウィザードでプロジェクトのフレームワークを作成します。次に、変換ウィザードで、選択したプロジェクトの種類に基づいてアプリケーションをコンパイルします。

Visual J++ プロジェクトを開くには

1. [ファイル] メニューの [プロジェクトを開く] をクリックします。
2. [プロジェクトを開く] ウィンドウで、変換するプロジェクト (.vjp) またはソリューション (.sln) を選択し、[開く] をクリックします。

   [Visual Studio 変換ウィザードへようこそ] ウィンドウが表示されます。
3. [次へ] をクリックします。
4. [バックアップの場所] フィールドで、アップグレード ウィザードが変換元のプロジェクトを格納するディレクトリを指定します。[次へ] をクリックします。

   変換するプロジェクトと同じディレクトリにバックアップを格納すると便利です。
5. [完了] をクリックします。

   Visual Studio 変換ウィザードによって、プロジェクトが自動的に Visual J# に変換され、変換のステータスが表示されます。
6. 変換中にエラーが発生したときは、リンクをクリックしてエラーと解説を確認します。

   または

   [閉じる] をクリックします。
7. 変換が完了したら、[プロジェクト] メニューの [プロジェクトの再読み込み] をクリックします。

   プロジェクトが新しい J# プロジェクトに更新され、J# アップグレード ウィザードが起動します。
8. [アップグレード ウィザードへようこそ] ページで、[次へ] をクリックします。
9. [プロジェクトの種類の選択] ページで、コンパイルするアプリケーションの種類として [コンソール アプリケーション]、[Windows アプリケーション]、または [クラス ライブラリ] を選択します。[次へ] をクリックします。
10. [参照をプロジェクトに追加する] ページで、[追加] をクリックし、アプリケーションが使用する .NET Framework ライブラリまたは COM ライブラリへのリンクを作成します。
11. [次へ] をクリックし、[完了] をクリックします。

    コンパイラによってアセンブリがビルドされ、bin\debug ディレクトリに配置されます。

参照

処理手順

方法 : 既存の J++ コードから Visual J# プロジェクトを作成する
方法 : Visual J++ 6.0 プロジェクトのアップグレード

その他の技術情報

Visual J++ 6.0 からのアップグレード

方法：J++/ActiveX アプリケーションをアップグレードする

Visual Studio 変換ウィザードを使用すると、J++ ActiveX プロジェクトを Visual J# プロジェクトに変換できます。プロジェクトを変換する前に、次の推奨事項について検討してください。

- アプリケーションをソース コードのリポジトリに格納しているときは、ファイルが使用できることを確認します。ウィザードは、変換中にファイルを自動的にチェックします。
- 必要なバックアップ ファイルをウィザードが作成できるようにします。変換が完了すると、プロジェクトやソリューションは以前のバージョンの Visual Studio には読み込まれません。

独自に作成したプロジェクトを変換する手順を次に示します。

プロジェクトをアップグレードするには

1. ファイルがリポジトリに存在する場合は、すべてのプロジェクト ファイルとアプリケーションのソース コードがチェックインされていることを確認します。
2. ソリューション エクスプローラで、[開く] をポイントし、[プロジェクト/ソリューション] をクリックします。
3. [プロジェクトを開く] ウィンドウで、アップグレードするプロジェクト ファイル (.vjp) またはソリューション ファイル (.sln) を選択し、[開く] をクリックします。

   Visual Studio 変換ウィザードが起動し、いったん変換が完了するとプロジェクトやソリューションは以前のバージョンの Visual Studio に読み込むことができなくなることが通知されます。
4. [次へ] をクリックします。
5. [バックアップ作成の選択] ページで、[変換前にバックアップを作成する] をクリックして変換元のプロジェクトのコピーを格納します。プロジェクトがソース コード管理されているため、追加のコピーが必要ない場合は、[作成しない] をクリックします。
6. [変換準備完了] ページで、タスクの概要一覧を確認します。必要に応じて、[前へ] ボタンを使用してバックアップ先を修正します。

   ソリューション、プロジェクト、またはコードがソース コード管理されている場合は、ウィザードによって適切なファイルがチェックアウトされ、必要に応じてバックアップが作成されます。ウィザードは、.vpn プロジェクト情報を新しい .vjproj プロジェクト ファイルに変換し、ソリューション ファイルを更新します。
7. [閉じる] をクリックします。
8. [ビルド] メニューの [ソリューションのビルド] をクリックして、アプリケーションを Visual J# アプリケーションに再コンパイルします。

参照

その他の技術情報

Java からの移行

方法 : VJ++ で記述された COM コンポーネントにアクセスする

共通オブジェクト モデル (COM: Common Object Model) オブジェクトは、機能をカプセル化し、複数のアプリケーション間で再利用するための有効な方法を提供します。COM オブジェクトを使用して、アプリケーション内の特定の機能を公開できます。たとえば、複数のアプリケーションで使用するルーチンのセットを作成できます。

COM は言語固有ではないため、COM オブジェクトは、さまざまなプログラミング言語で開発されたアプリケーションに簡単に取り込むことができます。Visual J# を使用して、システムに登録されている COM オブジェクトを表示したり、インポートしたりできます。インポート プロセス中には、Visual J# によってクラス ラッパーが作成されるため、他のオブジェクトと同じように COM オブジェクトにアクセスできます。

プロジェクトの COM コンポーネントへの参照を作成する方法を次の例に示します。

COM コンポーネントをインポートするには

1. ソリューション エクスプローラで、[参照設定] を右クリックし、[参照の追加] をクリックします。
2. [COM] タブをクリックして、登録済みの COM オブジェクトの一覧、またはその他のライブラリとコントロールの一覧を表示します。
3. インポートするコンポーネントを選択し、[OK] をクリックします。

   選択したコンポーネントが、ソリューション エクスプローラの参照の一覧に表示されます。
4. J# アプリケーションの COM コンポーネントにアクセスするには、適切な import (Visual J#) ステートメント (`import ClockComponent` など) を追加します。

参照

処理手順

方法 : COM コンポーネントを J# で生成する
方法 : COM オブジェクトを参照する

J# による Web プログラミング

Web ページの作成中には、ツールボックスを使用してフォームにグラフィカル コンポーネントを追加できます。ここでは、独自のコンポーネントを作成してツールボックスに追加する方法を示します。

このセクションの内容

方法 : ブラウザで実行されるコンポーネントを作成する

Web ページの開発用としてツールボックスに表示されるコンポーネントを作成する方法について説明します。

関連するセクション

方法 : VJ++ で記述された COM コンポーネントにアクセスする

Visual J# のタスク

方法：ブラウザで実行されるコンポーネントを作成する

コンポーネントは、**System.ComponentModel.Component** から継承される再利用可能なライブラリです。コンポーネントは、他のライブラリと同じ方法で使用できます。

ライブラリ コンポーネントを作成し、そのライブラリ コンポーネントを Web ページに追加し、Web サイトにコンポーネントを配置する方法を次の例に示します。Web ページ (.aspx) は、南アフリカにある企業オフィスの番地を要求します。このコンポーネントは、この要求を受け取ると、会社のケープタウン支社の住所を返します。

コンポーネントを作成するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックして [新しいプロジェクト] ダイアログ ボックスを開きます。
2. [新しいプロジェクト] ダイアログ ボックスで、[J# プロジェクト] の一覧の [クラス ライブラリ] プロジェクト テンプレートをクリックし、[プロジェクト名] ボックスに「**BrowserComponent**」と入力します。
3. [プロジェクト] メニューの [コンポーネントの追加] をクリックします。
4. [新しい項目の追加] ダイアログ ボックスの [コンポーネント クラス] をクリックし、[ファイル名] ボックスに「**BrowserComponent.jsl**」と入力します。

   BrowserComponent.jsl という名前のコンポーネントがクラス ライブラリに追加されます。
5. ソリューション エクスプローラで BrowserComponent.jsl を右クリックし、[コードの表示] をクリックします。

   コード エディタが表示されます。`System.ComponentModel.Component` が `public class BrowserComponent` の直後に表示されることに注目します。既定では、コンポーネントはシステムによって提供される **Component** クラスを継承します。**Component** クラスには、デザイナを使う機能を含め、コンポーネントのための多くの機能が用意されています。
6. ソリューション エクスプローラで、Class1.jsl を右クリックし、[削除] をクリックします。クラス ライブラリの既定のクラスが削除されます。このクラスは、この特定のコンポーネントには使用しません。
7. [ファイル] メニューの [すべてを保存] をクリックして、プロジェクトを保存します。
8. BrowserComponent.jsl に次のメソッドを追加します。

```
public String[] GetCompanyAddress(int CorpCode) throws IllegalArgumentException
    {
        String Address[];
        switch (CorpCode)
        {
            case 0:
                Address = new String[4];
                Address[0] = "1234 White Street";
                Address[1] = "Sturtevant";
                Address[2] = "WI";
                Address[3] = "53177";
                break;
            case 1:
                Address = new String[4];
                Address[0] = "oltke Moes vei 39";
                Address[1] = "0852";
                Address[2] = "OSLO";
                break;
            case 2:
                Address = new String[3];
                Address[0] = "Parliament Street";
                Address[1] = "Cape Town";
                Address[2] = "S. Africa";
                break;
            default:
                throw new IllegalArgumentException();
```

```
        }
        return Address;
    }
```

このメソッドは、架空の企業コードに基づいて、適切なカスタマー サービスの場所を指定します。**case** ステートメントは、返される内容を管理します。

9. [ビルド] メニューの [ソリューションのビルド] をクリックします。

   ライブラリがコンパイルを実行します。ただし、この時点では、このコンポーネントを呼び出すクライアントがないため、コンポーネントだけをコンパイルできます。

Web プロジェクトを作成するには

1. [ファイル] メニューの [追加] をポイントして、[新しい Web サイト] をクリックします。

   [新しい Web サイトの追加] ダイアログ ボックスが表示されます。

2. [場所] ボックスに「**C:\Inetpub\wwwroot\BrowserComponent**」と入力します。

3. [OK] をクリックします。

   プロジェクトによって、スタブ Web サイトが作成されます。Web.config ファイルが存在しないため、この時点ではビルドは実行されません。

4. Default.aspx を右クリックし、[スタート ページに設定] をクリックします。

5. Default.aspx を開き、内容を次のコードで置き換えます。

```
<%@ Import Namespace="BrowserComponent" %>

<html>

    <script language="VJ#" runat="server">

        public void Page_Load(Object sender, EventArgs E)
     {
            //Request the address in South Africa.
            BrowserComponent comp = new BrowserComponent();

            String[] arrAddress = comp.GetCompanyAddress(2);
            StringBuilder sbAddress = new StringBuilder();
            sbAddress.Append("<b>XYZ Customer Branch</b><br> ");
            for (int idx = 0; idx < arrAddress.length; idx++)
            {
                sbAddress.Append(arrAddress[idx]  + " <br> ");
            }
             Message.set_InnerHtml(sbAddress.ToString());
        }
    </script>
    <head><title>Locate Closest Customer Office</title></head>
    <body style="font: 10pt verdana">
        <h3>A Simple Managed Component</h3>
        <br><br><hr><br>
        <h5>Thank you for your inquiry. Our mailing address is:</h5>
        <table><tr><td>
        <div id="Message" runat="server" />
        </td></tr></table>
    </body>
</html>
```

Page_Load メソッドの **BrowserComponent** クラスに、新しいマネージ インスタンスが追加されています。このインスタンスは、**GetCompanyAddress** メソッドを呼び出して企業オフィス番号 291 の住所を要求します。拡張機能とし

て、**GetCompanyAddress** を呼び出す前に、独自のアプリケーションでユーザーにオフィス番号に関する情報の入力を求めることもできます。

Web ページを配置して表示するには

1. ソリューション エクスプローラで、Web サイト名を右クリックし、[参照の追加] をクリックします。
2. [参照の追加] ウィンドウの [プロジェクト] タブをクリックします。
3. [BrowserComponent] プロジェクトを強調表示し、[OK] をクリックします。

   後でソリューション全体をビルドする際に、参照によって Web ページがコンポーネントを見つけることができます。
4. ソリューション エクスプローラで、Web サイトの名前を右クリックし、[新しい項目の追加] をクリックします。
5. [Web 構成ファイル] をクリックし、[OK] をクリックします。

   Web.config ファイルは、セキュリティ モードなど、コンパイルおよび実行時の情報を提供します。
6. ソリューション エクスプローラで、web.config ファイルをダブルクリックします。
7. `<system.web>` の下に、次のコードを挿入します。

   ```
   <compilation debug="true" >
       <assemblies>
           <add assembly="BrowserComponent"/>
           </assemblies>

       </compilation>
   ```

   `compilation debug="true"` ステートメントは、Web ページからのコンポーネントのデバッグを許可します。Web サイトを稼動環境に配置する場合は、パフォーマンスを向上するため、このステートメントを false に設定してください。`add assembly="BrowserComponent"` 属性によって、Web ページが使用するコンポーネントが識別されます。
8. [ファイル] メニューの [すべてを保存] をクリックして、すべての参照とプロジェクトの設定を保存します。
9. [ビルド] メニューの [ソリューションのビルド] をポイントします。

   両方のプロジェクトでコンパイルが実行され、コンポーネントが Web サイトの \bin ディレクトリに配置されます。
10. [スタート] メニューの [プログラム] をポイントし、[管理ツール] をポイントして、[インターネット インフォメーション サービス] をクリックします。
11. [インターネット インフォメーション サービス] スナップインで、[BrowserComponent] フォルダを右クリックします。
12. [新規作成]、[仮想ディレクトリ]、[次へ] の順にクリックします。
13. [仮想ディレクトリ エイリアス] ウィンドウで、Web 名として「**BrowserComponent**」を入力します。
14. [次へ] をクリックします。
15. [参照] をクリックし、Web ディレクトリのパスを強調表示します。
16. [OK] をクリックし、[次へ] をクリックします。
17. [アクセス許可] ウィンドウで、[ISAPI アプリケーションや CGI などを実行する] チェック ボックスをオンにして BrowserControl.dll の実行を許可します。
18. [次へ] をクリックし、[完了] をクリックします。

    これで、Web サイトへのアクセスが有効になり、ブラウザで Web ページを実行できます。
19. Visual Studio で、[デバッグ] メニューの [開始] をクリックします。
20. ソリューション エクスプローラで、Default.aspx を右クリックし、[ブラウザで表示] をクリックします。

    Web ブラウザに、ケープタウンの住所が "XYZ Customer Branch Parliament Street Cape Town S. Africa" として表示されます。

参照

処理手順

方法 : J# ブラウザ コントロールを配置する

Visual J# による Windows GUI アプリケーションの作成

ここでは、Visual J# ライブラリを使用して Windows アプリケーションを作成する方法について説明します。

このセクションの内容

方法 : AWT のフレームを使用する

Abstract Window Toolkit (AWT) を使用して GUI を作成する方法について説明します。

方法 : 新しい Swing アプリケーションを作成する

Swing による GUI の作成方法について説明します。

関連するセクション

方法 : Visual J# Windows アプリケーションを作成する

Visual J# のタスク

方法：新しい Swing アプリケーションを作成する

Supplemental UI Library for Visual J# ライブラリは、従来のグラフィック ユーザー インターフェイス (GUI) とクライアント アプリケーションのグラフィックス機能の構築をサポートする Swing 機能のほとんどを提供します。vjssupuilib ライブラリは Visual J# に付属し、.NET Framework 内で実行される信頼性の高い Windows アプリケーションを構築するための機能を含んでいます。

次の例は、さまざまな Swing 機能とそれらを AWT ライブラリと相互運用できるしくみを示しています。また、この例では、Swing 機能をプログラムで作成する方法についても示します。このサンプルを実行すると、JPG グラフィックを読み込んで、イメージを網かけ表示し、形成し、任意の方向に配置できます。

フレーム スタブとメニュー スタブの作成

1. この例で編集するグラフィック .jpg ファイルを取得します。
2. Swing という名前の新しいコンソール プロジェクトを開きます。
3. ソリューション エクスプローラで、[参照設定] を右クリックし、[参照の追加] をクリックします。
4. VJSSupUILib.dll をクリックし、[OK] をクリックします。

   Microsoft Supplemental UI Library for Visual J# ライブラリへの参照がプロジェクトに追加されます。このライブラリには、ボタン、ボックス、境界線など、ユーザー アプリケーションの Swing コンポーネントが含まれます。
5. ソリューション エクスプローラで Program.jsl が選択されていることを確認し、コードを次のコードで置き換えます。

```
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.border.*;
import javax.swing.filechooser.FileFilter;
import java.awt.*;
import java.awt.image.*;
import java.awt.event.*;
import java.io.File;

public class ImageViewerApp extends JFrame implements ActionListener, ChangeListener
{
    // Component handles:
    private JMenuItem mLoad, mExit, mUndo, mOriginal,mProperties, mFlipX, mFlipY;
    private JSplitPane splitPane;
    private JLabel origImageLbl, workImageLbl;
    private JSlider sliderWidth, sliderHeight;

    // Working, original, and previous images:
    private Image workImage, originalImage, previousImage;
```

   クラス宣言が **JFrame** を継承して簡単な Swing ウィンドウを作成することに注目します。また、フレームを監視するために、この例では **ActionListener** インターフェイスと **ChangeListener** インターフェイスの両方を実装します。
6. 次のように、ビューアのメニューと書式設定を追加します。

```
    public ImageViewerApp(String title)
    {
        super(title);
        // Exit when window is closed.
        setDefaultCloseOperation(EXIT_ON_CLOSE);

        // Initalize each menu.
        initMenuBar();
        origImageLbl = new JLabel();
        workImageLbl = new JLabel();
```

```
        origImageLbl.setHorizontalAlignment(JLabel.CENTER);
        workImageLbl.setHorizontalAlignment(JLabel.CENTER);

        JScrollPane origImagePane = new JScrollPane(origImageLbl);
        origImagePane.setBorder(new TitledBorder("Original Image"));

        JScrollPane workImagePane = new JScrollPane(workImageLbl);
        workImagePane.setBorder(new TitledBorder("Modified Image"));

        splitPane = new JSplitPane();
        splitPane.setLeftComponent(origImagePane);
        splitPane.setRightComponent(workImagePane);
        splitPane.setDividerSize(10);
        splitPane.setOneTouchExpandable(true);
        splitPane.setContinuousLayout(true);
        getContentPane().add(splitPane);
        getContentPane().setBackground(Color.lightGray);

        setSize((int)(550 * 1.61), 550);
        setResizable(false);
        showWorkBench();
    }
```

このコードにより、画面の形状とサイズを手動で制御できます。たとえば、**setSize** メソッドの XY 座標を更新して画面のサイズを変更できます。

7. 次のように、メニュー バーに表示する各項目を定義し、ActionListener に追加します。

```
    private void initMenuBar()
    {
        JMenuBar menuBar = new JMenuBar();
        JMenu menu;

        // File.
        menu = new JMenu("File");
        menu.setMnemonic('F');
        menu.setToolTipText("Load Image to manipulate");

        mLoad = new JMenuItem("Load Image");
        mLoad.setMnemonic('L');
        mLoad.addActionListener(this);
        menu.add(mLoad);
        menu.addSeparator();

        mExit = new JMenuItem("Exit");
        mExit.setMnemonic('x');
        mExit.addActionListener(this);
        menu.add(mExit);
        menuBar.add(menu);

        // Edit.
        menu = new JMenu("Edit");
        menu.setMnemonic('E');
        menu.setToolTipText("Edit Image");

        mUndo = new JMenuItem("Undo");
        mUndo.setMnemonic('U');
        mUndo.setEnabled(false);
        mUndo.addActionListener(this);
```

```
        menu.add(mUndo);

        mOriginal = new JMenuItem("Revert to Original Image");
        mOriginal.setMnemonic('R');
        mOriginal.setEnabled(false);
        mOriginal.addActionListener(this);
        menu.add(mOriginal);

        mProperties = new JMenuItem("Scale");
        mProperties.setMnemonic('C');
        mProperties.addActionListener(this);
        mProperties.setEnabled(false);
        menu.add(mProperties);
        menuBar.add(menu);

        // Rotate.
        menu = new JMenu("Rotate");
        menu.setMnemonic('R');
        menu.setToolTipText("Rotate images");

        mFlipX = new JMenuItem("Flip Vertically");
        mFlipX.setMnemonic('V');
        mFlipX.setEnabled(false);
        mFlipX.addActionListener(this);
        menu.add(mFlipX);

        mFlipY = new JMenuItem("Flip Horizontally");
        mFlipY.setMnemonic('H');
        mFlipY.setEnabled(false);
        mFlipY.addActionListener(this);
        menu.add(mFlipY);
        menuBar.add(menu);
        setJMenuBar(menuBar);
    }
```

JMenuItem の各インスタンスは、ドロップダウン メニューに表示されるテキストを制御します。**setMnemonic** メソッドは、代替のショートカット キーを定義します。最後のステップで、**setJMenuBar** が各コンポーネントをメニュー バーに追加します。

8. 次のように、実行時に生成される **Workbench** というウィンドウを作成し、スライダ バーを実装し、**ActionListener** を実装します。

```
    // Shows the screen.
    private void showWorkBench()
    {
        final Window workBench;
        Timer timer = null;

        // Creates a Swing window, casts to AWT:
        JWindow wnd = new JWindow(this);
        wnd.pack();
        workBench = wnd;

        ActionListener listen = new ActionListener()
        {
            public void actionPerformed(ActionEvent event)
            {
                workBench.dispose();
                workBench.getParent().show();

                // Set center divider to 0.495 after frame displays.
```

```
                splitPane.setDividerLocation(0.495);
            }
        };
            // Center dialog on screen.
            Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
            workBench.setLocation((screenSize.width - workBench.getWidth()) / 2,
                (screenSize.height - workBench.getHeight()) / 2);

            // Listener is time-driven.
            // One second.
            timer = new Timer(1000, listen);
            timer.setRepeats(false);
            timer.start();
            workBench.show();
    }

    // Fires when size changes by slider.
    public void stateChanged(ChangeEvent event)
    {
        if (sliderWidth.getValueIsAdjusting() || sliderHeight.getValueIsAdjusting())
            return;

        int width = sliderWidth.getValue();
        int height = sliderHeight.getValue();

        Image filteredImage = workImage.getScaledInstance(width, height, Image.SCALE_S
MOOTH);
        setImage(filteredImage);
    }
```

`workBench = wnd` は、AWT **ActionListener** がウィンドウを管理できるように、**JWindow** を AWT **Window** にキャストすることに注目します。リスナは **Timer** を使用して、ウィンドウが 1 秒間隔で更新されているかどうかをチェックします。

9. 次のコードを挿入して各メニュー項目を実装します。

```
    public void actionPerformed(ActionEvent event)
    {
        Object evtSource = event.getSource();
        if (evtSource.equals(mUndo))
        {
            if (previousImage != null)
                setAndCommitImage(previousImage);
        }
        else if (evtSource.equals(mProperties))
            new ScaleDialog(this);
        else if (evtSource.equals(mOriginal))
        {
            setAndCommitImage(originalImage);
            // Restoring the original background:
            splitPane.getRightComponent().setBackground(
splitPane.getLeftComponent().getBackground());
        }
        else if (evtSource.equals(mLoad))
        {
            JFileChooser fileChooser = new JFileChooser(new File("."));
            fileChooser.setFileFilter(new JpgFileManager());

            int returnVal = fileChooser.showOpenDialog(this);
            if (returnVal != JFileChooser.APPROVE_OPTION)
```

```
                return;

            File file = fileChooser.getSelectedFile();
            if (file == null)
                return;
            else if (!file.exists())
            {
                String msg = "The file '" + file.getName() + "' not found.\n" +
                    "Please verify that the correct file name was given.";
                JOptionPane.showMessageDialog(this, msg, "Image Viewer",
                    JOptionPane.ERROR_MESSAGE);
                return;
            }

            if (workImage != null)
                workImage.flush();

            workImage = Toolkit.getDefaultToolkit().getImage(file.getAbsolutePath());
            if (workImage == null)
                return;

            originalImage = Toolkit.getDefaultToolkit().getImage(file.getAbsolutePath(
));
            origImageLbl.setIcon(new ImageIcon(originalImage, ""));
            workImageLbl.setIcon(new ImageIcon(workImage, ""));
            previousImage = null;
            mFlipX.setEnabled(true);
            mFlipY.setEnabled(true);
            mUndo.setEnabled(false);
            mOriginal.setEnabled(true);
            mProperties.setEnabled(true);
            repaint();
        }

        else if (evtSource.equals(mExit))
        {
            dispose();
            System.exit(0);
        }
        (evtSource.equals(mFlipX))
            flip(false);
        else if (evtSource.equals(mFlipY))
            flip(true);
    }
```

event.getSource メソッドがユーザーのアクションを返すことに注目します。アプリケーションにさらに機能を追加する場合は、**else if** 句を追加してこれらのイベントを管理します。

10. アプリケーションを呼び出す **main** メソッドを挿入します。

```
    public static void main(String[] args)
    {
        new ImageViewerApp("ImageViewer - A Visual J# .NET Demo");
    }
```

11. [ビルド] メニューの [ソリューションのビルド] をクリックします。

12. この時点でアプリケーションを実行すると、メニュー項目を持つ空のウィンドウが開きます。

読み込みを管理するためのコードを追加するには

- 次のコードを挿入して、JPG グラフィックスを使用するユーザーを支援します。

```
    class JpgFileManager extends FileFilter
    {
        public String getDescription()
        {
            return "JPG Files (*.jpg)";
        }
// Seeks directory or file.
        public boolean accept(File file)
        {
            if (file.isDirectory())
            {
                return true;
            }
            else
            {
                String filepathname = file.getAbsolutePath().toLowerCase();
                if (filepathname.endsWith(".jpg"))
                    return true;
            }
            return false;
        }
    }
//Rotates the image and copies it to the right panel.
    private void flip(boolean aboutY)
    {
        final int width = workImage.getWidth(null);
        final int height = workImage.getHeight(null);
        final int total = width * height;

        // Load the workImage pixels into the imagePixels array:
        int imagePixels[] = new int[total];

        PixelGrabber pg = new PixelGrabber(workImage, 0, 0, width, height,
            imagePixels, 0, width);
        try
        {
            pg.grabPixels();
        }
        catch (InterruptedException e) { } ;

        if (aboutY)
        {
            // About Y-axis:
            int center = width / 2;
            int temp;
            for (int t = 0; t < total; t += width)
            {
                for (int c = 0; c < center; c++)
                {
                    temp = imagePixels[t + c];
                    imagePixels[t + c] = imagePixels[t + width - 1 - c];
                    imagePixels[t + width - 1 - c] = temp;
                }
            }
        }
```

```
        else
        {
            // About X-axis:
            int center = total / 2;
            int temp;
            for (int t = 0; t < width; t++)
            {
                for (int c = 0; c < center; c += width)
                {
                    temp = imagePixels[t + c];
`                    imagePixels[t + c] = imagePixels[total + t - width - c];
                    imagePixels[total + t - width - c] = temp;
                }
            }
        }

        Image filteredImage = createImage(new
 MemoryImageSource(width, height, imagePixels, 0, width));
        setAndCommitImage(filteredImage);
    }
// Copies the image to the right panel that stores modified image.
    private void setImage(Image newImage)
    {
        workImageLbl.setIcon(new ImageIcon(newImage));
    }
// Refresh the right panel that stores modified image.
    private void setAndCommitImage(Image newImage)
    {
        if (workImage != null)
            workImage.flush();

        previousImage = workImage;
        workImage = newImage;
        workImageLbl.setIcon(new ImageIcon(workImage));
        mUndo.setEnabled(true);
    }
    private void commitImage()
    {
        Icon icon = workImageLbl.getIcon();
        setAndCommitImage(((ImageIcon)icon).getImage());
    }
}
```

ユーザーが特定のディレクトリにイメージを読み込むのを支援するため、**JpgFileManager** クラスが **FileFilter** を拡張していることに注目します。**flip** メソッドでは、イメージが回転し、**ActionListener** を呼び出して、変更されたイメージを格納する右パネルにグラフィックをコピーします。

グラフィックをスケーリングするコードを追加するには

1. 次のクラスを追加してスケーリングを管理します。次のコードを挿入してください。

```
class ScaleDialog extends JDialog implements ActionListener
    {
        public ScaleDialog(JFrame parent)
        {
            super(parent, "Scale", true);

            int width = workImage.getWidth(null);
            int height = workImage.getHeight(null);
```

```
        int maxW = Math.max(width, 1200);
        int maxH = Math.max(height, 1000);

        sliderWidth = new JSlider(0, maxW, width);
        sliderWidth.setBorder(new TitledBorder("Width"));
        sliderWidth.setMajorTickSpacing(100);
        sliderWidth.setPaintTicks(true);
        sliderWidth.setPaintLabels(true);
        sliderWidth.setMinorTickSpacing(50);
        sliderWidth.addChangeListener((ChangeListener)parent);

        sliderHeight = new JSlider(0, maxH, height);
        sliderHeight.setBorder(new TitledBorder("Height"));
        sliderHeight.setMajorTickSpacing(100);
        sliderHeight.setPaintTicks(true);
        sliderHeight.setPaintLabels(true);
        sliderHeight.setMinorTickSpacing(50);
        sliderHeight.addChangeListener((ChangeListener)parent);

        JPanel sliders = new JPanel(new GridLayout(2, 1, 4, 4));
        sliders.add(sliderWidth);
        sliders.add(sliderHeight);

        // ok cancel buttons panel
        JButton ok = new JButton("OK");
        JButton cancel = new JButton("Cancel");
        ok.addActionListener(this);
        cancel.addActionListener(this);
        ok.addActionListener((ActionListener)parent);
        cancel.addActionListener((ActionListener)parent);

        JPanel buttonsPanel = new JPanel();
        buttonsPanel.add(ok);
        buttonsPanel.add(cancel);

        getContentPane().add(sliders, BorderLayout.CENTER);
        getContentPane().add(buttonsPanel, BorderLayout.SOUTH);

        setResizable(false);
        setSize((int)(300 * 1.61), 300);
        setLocationRelativeTo(parent);

        // Add a Window closing listener to simulate cancel.
        addWindowListener(new WindowAdapter()
        {
            public void windowClosing(WindowEvent e)
            {
                setImage(workImage);
            }
        });

        show();
    }

    public void actionPerformed(ActionEvent event)
    {
        String eventName = (String)event.getActionCommand();
        if (eventName.compareTo("OK") == 0)
        {
```

```
                commitImage();
                dispose();
            }
            else if (eventName.compareTo("Cancel") == 0)
            {
                setImage(workImage);
                dispose();
            }
        }
    }
```

ScaleDialog コンストラクタは親ウィンドウを受け入れ、イメージをスケーリングするためのスライダ ボックスを作成します。新しいスライダ バーに手動で値を入力する代わりに、**setMajorTickSpacing(100)** メソッドにより 100 目盛りごとに値を挿入します。別の **actionPerformed** メソッドがスライダの [OK] ボタンと [キャンセル] ボタンを処理することに注目します。

2. F5 キーを押してアプリケーションをコンパイルおよび実行します。

   イメージ編集のためのウィンドウが開きます。グラフィックを開くと、イメージが両方のラベル ボックス パネルに表示されます。グラフィックを編集したり向きを変更したりすると、更新内容が右パネルに反映されます。

参照

処理手順

方法 : AWT のフレームを使用する

方法：AWT のフレームを使用する

java.awt パッケージには、Visual J# Abstract Windowing Toolkit (AWT) を構成するクラスおよびサブパッケージが含まれます。AWT は、Java で記述された GUI アプリケーションをすべてのプラットフォームで実行できるようにするレガシ ユーザー インターフェイス オブジェクトを提供します。**java.awt** パッケージに含まれるクラスの例には、**Button**、**List**、**Menu**、**Checkbox**、**Color** などがあります。

AWT アプリケーションは Windows フォームを使用しないため、コンソール アプリケーション テンプレートを使用してプロジェクトを作成できます。AWT を使用して Microsoft® メモ帳スタイルの独自のエディタ アプリケーションを作成する方法を次の例にします。この例では、デザイナにドラッグする代わりに、プログラムを使用してウィンドウとコンポーネントを作成する方法を示します。

簡易テキスト エディタを作成するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。
2. J# の [プロジェクトの種類] ボックスの一覧の [コンソール アプリケーション] テンプレートをクリックし、[名前] ボックスに「AWT」と入力します。
3. ソリューション エクスプローラで Program.jsl が選択されていることを確認し、コード エディタのコードを次のコードで置き換えます。

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;

class Editor extends Frame implements ActionListener
{
    TextArea textArea = new TextArea();

    //Set up the menu on the default CTOR.
    Editor()
    {
    super("Text AWT Editor");
        setLayout(new BorderLayout());
        add("Center", textArea);
        Menu menu = new Menu("File");
        menu.add(makeMenuItem("Open"));
        menu.add(makeMenuItem("Save"));
        menu.add(makeMenuItem("Quit"));
        MenuBar menuBar = new MenuBar();
        menuBar.add(menu);
        setMenuBar(menuBar);
        pack();
    }
    private MenuItem makeMenuItem(String name)
    {
        MenuItem m = new MenuItem(name);
        m.addActionListener(this);
        return m;
    }
    public static void main(String[] s)
    {
        new Editor().show();
    }
```

super キーワードは、`super("Text AWT Editor");` 行のフレームにタイトル バーを設定します。フレーム レイアウトは **BorderLayout** に設定されます。この設定により、コンポーネント間にすきまのないウィンドウが作成されます。その他のフレーム形式には、**CardLayout**、**GridBagLayout** などがあります。

新しい `Menu("File")` メソッドと **menu.add** メソッドは、ユーザーが選択する項目を含むドロップダウン メニューを作成します。

4. 次のコードを挿入して、各ドロップダウン メニュー項目のイベント ハンドラを実装します。

```
    public void actionPerformed(ActionEvent e)
    {
        String command = e.getActionCommand();
        if (command.equals("Quit"))
            dispose();
        else if (command.equals("Open"))
            openFile();
        else if (command.equals("Save"))
            saveFile();
    }
```

5. 次のコードを挿入して、ファイルを開く操作および保存する操作を管理するメソッドを実装します。

```
private void openFile()
    {
        //Show the Open File dialog box to the user.
        FileDialog fd = new FileDialog(this, "Open File", FileDialog.LOAD);
        fd.show();

        //Get the file path.
        StringBuffer sbPath = new StringBuffer(fd.getDirectory());
        sbPath.append("\\");
        String fileName = fd.getFile();

        //Cancel if null:
        if (fileName == null)
        {
            return;
        }
        else sbPath.append(fileName);

        // Open and fill the input stream; paint the TextArea.
        try
        {
            FileInputStream fs = new FileInputStream(sbPath.ToString());
            byte[] data = new byte [ sbPath.length() ];
            fs.read(data);
            textArea.setText(new String(data));
        }
        catch (IOException e)
        {
            textArea.setText(e.toString());
        }
    }

    private void saveFile()
    {
        try
        {
            // Show the dialog box to the user.
            FileDialog fd = new FileDialog(this, "Save File", FileDialog.SAVE);
            fd.show();

            //Get the path and file name.
            StringBuffer sbPath = new StringBuffer(fd.getDirectory());
            sbPath.append("\\");
            // Get the file name to create:
```

```
                sbPath.append(fd.getFile());
                FileOutputStream fo = new FileOutputStream(sbPath.ToString());

                // Parse the data:
                String strdata = textArea.getText();
                byte[] data = new byte [strdata.length()];
                for (int idx = 0; idx < strdata.length(); idx++)
                {
                    data[idx] = (byte)strdata.charAt(idx);
                }
                // Write the array to the file.
                fo.write(data);
            }
            catch (IOException ex)
            {
                textArea.setText(ex.toString());
            }
        }
    }
```

6. F5 キーを押してアプリケーションをコンパイルおよび実行します。

   テキスト エディタが開き、文字を入力できるようになります。入力後には、作業を保存したり、別のテキスト ファイルを開いて編集したりできます。

参照

処理手順

方法 : 新しい Swing アプリケーションを作成する

方法：.properties ファイルからリソースにアクセスする

プロジェクトを Visual J# に変換すると、.properties ファイルと .txt ファイルのデータは、自動的に .resx という種類のファイルにアップグレードされます。ただし、レガシ アプリケーションと新しい J# アプリケーションで同じプロパティ ファイルを共有する必要がある場合は、.properties ファイルにアクセスできます。

使用例

次の例は、従来の .properties ファイルに対する読み書きの方法を示しています。コードは、.properties ファイルを作成して格納し、その内容をフェッチして出力します。

```
package Properties;
import java.util.*;
import java.io.*;
/**
 * Summary description for Program
 */
public class Program
{
    public static void main(String[] args)
    {
        try {
            // Build a properties file for student.
            File file = new File("c:\\MyResources.properties");
            file.createNewFile();
            Properties properties = new Properties();
            properties.load(new FileInputStream(file));
            properties.setProperty("CountryName", "South Africa");
            properties.setProperty("CountryCode", "27");
            properties.setProperty("Localization", "afr");
            properties.setProperty("AccessLevel", "Student");
            properties.setProperty("CityName", "Johannesburg");
            properties.setProperty("CityCode", "11");
            Enumeration e = properties.keys();

            while (e.hasMoreElements())
            {
                Object obj = e.nextElement();
                System.out.println(obj.toString());
                System.out.println(properties.getProperty(obj.toString()));
            }
        } catch (IOException ex) {
                System.out.println(ex.toString());
        }
      }
}
```

コードのコンパイル方法

- F5 キーを押してアプリケーションをコンパイルおよび実行します。

方法：Visual J# から DLL のメソッドを呼び出す

アプリケーションの開発時には、できる限り既存のライブラリ コードを使用することをお勧めします。再利用できるコードを使用することにより、機能を自分で記述する必要がなくなります。

DLL から関数 (メソッド) を呼び出す方法は 2 つあります。静的メソッドにアクセスするか、インスタンスを作成してから動的にメソッドを呼び出します。vjslib.dll ライブラリから静的メソッドおよび動的メソッドを呼び出す方法を次の例に示します。

DLL からメソッドを呼び出すには

1. コンソール アプリケーションを作成し、**CallDll** という名前を付けます。
2. Program.jsl の内容を次のコードで置き換えます。

```
import java.io.*;
public class MyClass
{
    public String CreateFile()
    {
        // This is a call to a static method.
        System.out.println("Because there is not instance, this is a call to a static
method");

        // This is a call to a dynamic method because
        //there is a File instance.
        File myFile = new File("c:\\temp.txt");
        return myFile.getPath();
    }

    public static void main(String[] args)
    {
        MyClass myClass = new MyClass();
        String OutputFile = myClass.CreateFile();
        System.out.println("The following file was created: " + OutputFile);
    }
}
```

   CreateFile メソッドは、まず、**System.Console.WriteLine** を静的なメソッドとして呼び出します。**WriteLine** メソッドを使用するために、**System.Console** をインスタンス化する必要はありません。ただし、ファイルを作成するには、**File** クラスにマネージ メモリを割り当てるために **new** キーワードが必要です。たとえば、`File myFile = new File("c:\\temp.txt")` となります。

   ネイティブではない外部のライブラリについては、プロジェクト内から DLL への参照の追加が必要になる場合があります。コンパイラがライブラリを見つけることができなかった場合、コードで使用しているメソッド名の下に緑の波線が表示されます。

3. ソリューション エクスプローラで、[参照の追加] を右クリックし、[参照] をクリックして、使用するクラスおよびメソッドを含むライブラリ DLL を見つけます。
4. F5 キーを押してアプリケーションをコンパイルおよび実行します。

   ネイティブではない外部のライブラリについては、プロジェクト内から DLL への参照の追加が必要になる場合があります。コンパイラがライブラリを見つけることができなかった場合、コードで使用しているメソッド名の下に緑の波線が表示されます。

5. ライブラリ名をクリックし、[OK] をクリックします。
6. F5 キーを押してアプリケーションをコンパイルおよび実行します。

   CallDll が起動して **CreateFile** メソッドが実行されると、コンソールに "`This is a call to a static method`" と表示され、c:\\temp.txt という名前のファイルが作成されます。

コードのコンパイル方法

- ライブラリ メソッドを呼び出すには、完全修飾名を使用するか、コードの最初の **import** ステートメントでライブラリ パッケージ名を指定します。.jsl ファイルの先頭で import (Visual J#) ステートメントを使用してください。
- クラスを継承しない場合は、ライブラリ メソッドを **public** にする必要があります。継承する場合は、クラスの **public** メンバと **protected** メンバにアクセスできます。
- メソッドが static (Visual J#) でない場合は、クラスのインスタンスを通じて参照される必要があります。クラスのインスタンスを取得するには、new (Visual J#) キーワードを使用します。
- プロパティや変数に式を割り当てる場合は、戻り値を使用するか、互換性のあるデータ型にキャストします。たとえば、**int** を返す `CountPeople()` というライブラリ メソッドからの戻り値を使用する場合は、データ型も **int** である必要があります。例 : `int Total = CountPeople();`

セキュリティ

不明なソースに属するライブラリを使用する場合は、データの入出力が有効であることを検査します。

参照

その他の技術情報

クラス ライブラリのサポート

方法：既存の J++ コードから Visual J# プロジェクトを作成する

Visual J# アップグレード ウィザード を使用して、J++ アプリケーションを .NET Framework にアップグレードできます。プロジェクト全体を Visual J# にアップグレードするための所要時間はわずか 1 分です。Visual J++ プロジェクトを Visual J# にアップグレードする方法を次の例に示します。

Visual J++ 6.0 プロジェクトを Visual J# にアップグレードするには

1. [ファイル] メニューの [開く] をクリックし、[プロジェクト/ソリューション] をクリックします。

   [プロジェクトを開く] ダイアログ ボックスにプロジェクトの一覧が表示されます。

2. VJ++ プロジェクトをポイントし、[OK] をクリックします。

   Visual J# アップグレード ウィザード によって、Visual J++ プロジェクトが自動的に変換されます。アップグレードするプロジェクトに関連付けられているリソース ファイル (拡張子が .resources と .properties のファイル) も .NET Framework XML リソース形式 (.resx) ファイルに変換されます。ただし、ビルド前のステップおよびビルド後のステップはサポートされていません。

3. [ビルド] メニューの [ソリューションのビルド] をクリックします。

   CLASSPATH を設定する必要はありません。これは、Visual J# によって自動的に管理されます。jsc コンパイラによってコードがコンパイルされ、プロジェクトの bin\debug ディレクトリにバイナリが配置されます。また、すべてのバージョン情報はアセンブリ内に属性として存在します。

参照

処理手順

方法 : Visual J++ プロジェクトを変換する
方法 : Visual J# 再頒布パッケージのバージョン管理を設定する

方法：J# ブラウザ コントロールを配置する

Microsoft J# Web ブラウザ コントロールは、1 つ以上のグラフィカル ユーザー コントロールを含むライブラリです。このコントロールがアセンブリ DLL に含まれる場合、Web 開発者はアセンブリを参照し、ツールボックスから Web ページにコントロールをドラッグできます。

コントロールを含むプロジェクトと Web プロジェクトの両方をコンパイルすると、ブラウザ コントロール DLL が自動的に Web サイトに配置されます。ブラウザ コントロール DLL だけが存在する場合は、Web サイトの bin ディレクトリ (c:\inetpub\wwwroot\mywebsite\bin など) にブラウザ コントロール DLL を手動でコピーする必要があります。

単純なコントロールを作成し、そのコントロールを Web サイトに配置し、Web ページ上に表示する方法を次の例に示します。

ブラウザ コントロールの作成

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックして [新しいプロジェクト] ダイアログ ボックスを開きます。
2. [J# プロジェクト] ボックスの一覧の [Web コントロール ライブラリ] プロジェクト テンプレートを選択し、[プロジェクト名] ボックスに「**BrowserControl**」と入力します。
3. WebCustomControl1.jsl のコードを次のコードで置き換えます。

```
package MyTextBox;

import System.Web.UI.*;
import System.Web.UI.WebControls.*;
import System.ComponentModel.*;

/**
 * Summary description for WebCustomControl1.
 */
/**
 * @attribute DefaultProperty("Text")
 * @attribute ToolboxData("<{0}:WebCustomControl1 runat=server></{0}:WebCustomControl1
>")
 */
public class MyTextBox extends System.Web.UI.WebControls.WebControl
{
    private String text = "";
    private boolean Visible = false;
    /**
     * @attribute Bindable(true)
     * @attribute Category("Appearance")
     * @attribute DefaultValue("")
     */


    public String get_Text()
    {
        return text;
    }

    public void set_Text(String strInput)
    {
        text = strInput;
    }

    /**
     * Render this control to the output parameter specified.
     */
    protected void Render(HtmlTextWriter output)
```

```
    {
        output.Write(text);
    }
    public void SetVisible(boolean bShow)
    {
        Visible = bShow;
    }
    public boolean GetVisible()
    {
        return Visible;
    }
     // 3a: Postback -> LoadViewState

   protected void LoadViewState(Object viewState) {
    //Page.Trace.Write (" -> LoadViewState");
    if (viewState == null) {
     LoadViewState(viewState);
    }
   }

}
```

SetVisible メソッドと **GetVisible** メソッドが実行時のコントロールの表示を管理していることに注目します。

@attribute フラグは、デザイン時に Web 開発者に対して表示される既定のプロパティを制御します。

4. [ファイル] メニューの [すべてを保存] をクリックして、すべての参照とプロジェクトの設定を保存します。
5. [ビルド] メニューの [ソリューションのビルド] をクリックします。

   ライブラリがコンパイルを実行します。ただし、この時点では、このコンポーネントを実際に呼び出すクライアントがないため、何も実行されません。コンポーネントのコンパイルのみが行われます。

コントロールへの Web ページの配置

1. [ファイル] をクリックし、[追加] をポイントして、[新しい Web サイト] をクリックします。
2. [新しい Web サイト] ダイアログ ボックスの [場所] ボックスをクリックし、「**C:\Inetpub\wwwroot\BrowserControl**」と入力します。
3. [OK] をクリックします。

   プロジェクトによって、スタブ Web サイトが作成されます。Web.config ファイルが存在しないため、この時点ではビルドは実行されません。
4. ソリューション エクスプローラで、Default.aspx を右クリックし、[スタート ページに設定] をクリックします。

   [表示] メニューの [ツールボックス] をクリックします。

   ツールボックスが開き、BrowserControls という新しいコントロール セットが表示されます。
5. ツールボックスから、フォームに Button をドラッグします。
6. [プロパティ] ウィンドウで、Text プロパティを「**Get Balance**」に変更します。
7. ツールボックスで、[BrowserControls] をポイントし、クリックしてコントロールの一覧を開きます。
8. [MyTextBox] ブラウザ コントロールを Default.aspx ページにドラッグします。

   コントロールがブラウザに表示され、コントロールの作成時にエラーが発生したことを示すメッセージが表示されます。このエラーは、コントロールを構成する必要があることを通知しているだけです。
9. 新しいコントロールをダブルクリックします。

   コード ウィンドウに default.aspx Web ページが表示されます。
10. `cc1` カスタム コントロール タグの横にコントロールのクラス名を入力します。開始タグと終了タグの両方に入力してください。たとえば、次のようにします。

```
<cc1:MyTextBox id="WebCustomControl1_1" runat="server" ForeColor="Black" Width="258px"
Height="15px" Text="333"></cc1:MyTextBox>
```

11. [デザイン] タブをクリックしてデザイナを表示します。
12. コントロールを右クリックし、[プロパティ] をクリックします。
13. Text プロパティに、「**Balance**」と入力します。

    背景色や前景色など、他のプロパティを設定することもできます。

ブラウザ コントロールの配置と実行

1. ソリューション エクスプローラで、Web サイト名を右クリックし、[参照の追加] をクリックします。
2. [参照の追加] ウィンドウの [プロジェクト] タブをクリックします。
3. [BrowserControl] プロジェクトを選択し、[OK] をクリックします。

    後でソリューションをビルドする際に、参照によって Web ページがコンポーネントを見つけることができます。
4. ソリューション エクスプローラで、Web サイトの名前を右クリックし、[新しい項目の追加] をクリックします。
5. [Web 構成ファイル] をクリックし、[OK] をクリックします。

    Web.config ファイルは、セキュリティ モードなど、コンパイルおよび実行時の情報を提供します。
6. ソリューション エクスプローラで、web.config ファイルをダブルクリックします。
7. `<system.web>` の下に、次のコードを追加します。

```
        <compilation debug="true" >
            <assemblies>
                <add assembly="BrowserControl"/>
                </assemblies>

            </compilation>
```

    `compilation debug="true"` ステートメントは、Web ページからのコンポーネントのデバッグを許可します。Web サイトを稼動環境に配置する場合は、パフォーマンスを向上するため、このステートメントを `false` に設定してください。`add assembly="BrowserControl"` 属性によって、Web ページが使用するコンポーネントが識別されます。
8. [ファイル] メニューの [すべてを保存] をクリックして、すべての参照とプロジェクトの設定を保存します。
9. [ビルド] メニューの [ソリューションのビルド] をクリックします。

    両方のプロジェクトでコンパイルが実行され、コンポーネントが Web サイトの \bin ディレクトリに配置されます。
10. [スタート] メニューの [プログラム] をポイントし、[管理ツール] をポイントして、[インターネット インフォメーション サービス] をクリックします。
11. [インターネット インフォメーション サービス] スナップインで、[BrowserControl] フォルダを右クリックします。
12. [新規作成]、[仮想ディレクトリ]、[次へ] の順にクリックします。
13. [仮想ディレクトリ エイリアス] ウィンドウの [Web 名] ボックスに「**BrowserControl**」を入力します。
14. [次へ] をクリックします。
15. [参照] をクリックし、Web ディレクトリのパスをクリックして Web サイトの場所を指定します。
16. [OK] をクリックし、[次へ] をクリックします。
17. [アクセス許可] ウィンドウで、[ISAPI アプリケーションや CGI などを実行する] チェック ボックスをオンにして BrowserControl.dll の実行を許可します。
18. [次へ] をクリックし、[完了] をクリックします。

    これで、Web サイトへのアクセスが有効になり、ブラウザで Web ページを実行できます。

19. Visual Studio で、[デバッグ] メニューの [開始] をクリックします。
20. ソリューション エクスプローラで、Default.aspx を右クリックし、[ブラウザで表示] をクリックします。
21. [Balance] をクリックします。

    TextBox ブラウザ コントロールが "Balance = $497.22" に更新されます。独自に作成したアプリケーションでは、コントロールの背後にロジックを追加して、ユーザーの入力に応じて異なる出力を表示できます。

参照

処理手順

方法 : ブラウザで実行されるコンポーネントを作成する

方法：J# にインターフェイスを実装する

インターフェイスは、プロパティ、メソッド、およびイベントの特性を記述するだけでなく、クラスにより、これらのオブジェクトの動作をカスタマイズできるようにします。implements キーワードは、インターフェイスと同じシグネチャを持つメソッドをクラスで宣言し、実装できるようにします。

たとえば、**Car** インターフェイスは、**Color** というプロパティと **Drive()** というメソッドを持つことができます。クラスでは、**Car** インターフェイスを実装し、**Color** プロパティで乗り物を説明し、**Drive()** メソッドを実装して乗り物の動きを記述できます。

インターフェイスを実装する場合は、次の点を考慮します。

- クラスのメソッドは親インターフェイスと同じシグネチャを持つ必要があります。たとえば、メソッドは、インターフェイスと同じ数と種類のパラメータを持ち、同じ戻り値の型を持つ必要があります。
- vjslib.dll 内のクラスの実装をオーバーライドできますが、一定の制限があります。たとえば、**java.lang.Object** や **java.lang.String** の実装はオーバーライドできません。また、例外クラスはいずれもオーバーライドできません。

銀行の取引を比較する **java.lang.Comparable** インターフェイスを実装する方法を次の例に示します。

インターフェイスを実装するには

1. 新しいコンソール アプリケーションを開き、Verify Transaction という名前を付けます。
2. 次のクラスを .jsl ファイルにコピーします。

```
public class SecurityAudit implements java.lang.Comparable
{
    String AccountNumber;
    Double AccountBalance;
    String ReferenceNumber;

    public int compareTo(Object Verify)
    {

        final int SameTransaction = 0;
        final int SuspiciousTransaction = 1;

        if (this == Verify)
            return SameTransaction;
        else return SuspiciousTransaction;
    }
```

compareTo メソッドは、2 つのオブジェクトを比較する独自のコードを持つインターフェイスを実装します。メソッド シグネチャや戻り値の型は変更できません。

3. **main** メソッドを次のコードで置き換えます。

```
public static void main(String[] args)
{
        SecurityAudit Original = new SecurityAudit();
        Original.AccountNumber = "987-654";
        Original.AccountBalance = new Double(777.77);
        Original.ReferenceNumber = "1234";

        SecurityAudit CustomerClaim = new SecurityAudit();
        CustomerClaim.AccountNumber = "987-654";
        CustomerClaim.AccountBalance = new Double(977.00);
        CustomerClaim.ReferenceNumber = "1234";
        CustomerClaim.compareTo(Original);
        if (Verify == 0)
            System.out.println("Reference Number" + CustomerClaim.ReferenceNumber + "
```

```
appears valid");
        else
            System.out.println("Reference Number" + CustomerClaim.ReferenceNumber + "
does not match the original reference. Contact your supervisor");
}
```

main メソッドは 2 つの銀行取引をシミュレートします。インスタンス化時に、このコードは各銀行取引の **SecurityAudit** オブジェクトをインスタンス化し、**compareTo** メソッドを呼び出します。

4. F5 キーを押してアプリケーションをコンパイルおよび実行します。

   compareTo メソッドは両方のオブジェクトを比較し、残高が一致しないことを報告します。アプリケーションは "`Reference Number1234 does not match the original reference. Contact your supervisor`" と報告します。**compareTo** は **int** を返すため、このメソッドにより銀行取引に関するさまざまな問題を追跡できます。

参照

処理手順

方法 : Visual J# から DLL のメソッドを呼び出す

方法：コンソール アプリケーションへのユーザー入力を検証する

入力パラメータ フラグを受け入れるコンソール アプリケーションを作成できます。起動時にユーザーがコンソール アプリケーションに入力すると、共通言語ランタイム (CLR: Common Language Runtime) は空白区切りの入力パラメータの配列を **main** メソッドに提示します。

これらのパラメータを使用して、プログラムの動作を制御できます。実行時に、ユーザーが姓名を入力したことを確認するコンソール アプリケーションを次の例に示します。Maxlength フィールドによって、ユーザーが 20 文字を超える架空の名を入力したときに、プログラムは確実に IllegalArgumentException をスローします。独自のアプリケーションでは、姓の文字数の検査など、追加のチェックを指定することもできます。

ユーザー入力を検証するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。
2. [コンソール アプリケーション] をクリックし、[プロジェクト名] ボックスに「**ValidateUserInput**」と入力します。
3. Program.jsl のコードを次のコードで置き換えます。

```
package ValidateUserInput;

public class Program
{
    public static void main(String[] args)
    {
        int MaxLength = 20;
        if (args.length != 2)
        {
            System.out.println("Usage: ValidateUserInput FirstName LastName ");
            System.out.println("For example:");
            System.out.println("ValidateUserInput John Doe");
        }
        String input = new String(args[1]);
        if (input.get_Length() > MaxLength)
             throw new IllegalArgumentException();

    }
}
```

4. ソリューション エクスプローラで、プロジェクト名 ValidateUserInput を右クリックし、[プロパティ] をクリックします。
5. [デバッグ] をクリックします。
6. [コマンド ライン引数] ボックスに、コマンド プロンプトで通常使用するコマンド ライン引数を入力します (例 : **John Doe**)。
7. F5 キーを押して、プログラムをコンパイルおよび実行します。

   プログラムがデバッグ モードで実行されているときに、ValidateUserInput というアプリケーションが自動的に起動し、2 つの入力パラメータ (*John Doe* など) が入力されます。コンソール ウィンドウからアプリケーションを実行する場合は、「**ValidateUserInput John Doe**」と入力する必要があります。

セキュリティ

バッファ オーバーランや悪意による攻撃を防ぐため、コードでは、ユーザー入力の内容と文字数を常に検査する必要があります。

参照

関連項目

Visual J# で記述されたアプリケーションでのセキュリティのセマンティクス

方法：package スコープのクラスを記述する

package ステートメントを使用して、アプリケーション内の関連するクラスをグループ化できます。同じパッケージ名を持つクラスは package スコープのクラスと呼ばれます。スコープを設定すると、同じパッケージ内のクラス メンバを相互に使用できます。また、クラスの呼び出し元は、簡単な **import** ステートメントを使用してスコープ全体にアクセスできます。

パッケージを使用する場合は、次の点を考慮します。

- 複数の単語から成るパッケージ名を検討します。名前付け規則によって、最初の単語は組織名を示します。たとえば、**package mycompany** パッケージのようになります。
- パッケージを入れ子にするには、ドット (区切り記号) と子の名前を追加します。たとえば、**mycompany.platform.security** のようになります。
- 設計上、パッケージのスコープはアセンブリ間 (.exe および .dll ファイル) をまたいで発生します。アセンブリへのアクセスをブロックする必要がある場合は、/securescoping コンパイラ スイッチを使用します。

package スコープのクラスの使用方法を次の例に示します。新しい Visual J# プロジェクトを作成する場合、プロジェクト名が自動的にプロジェクト内の各クラスのパッケージ名になります。既存のクラスの場合、**package** ステートメントを手動で追加または変更する必要があります。

新しいアプリケーションを自動的にパッケージにグループ化するには

1. コンソール アプリケーションを作成し、MyPackage という名前を付けます。
2. Program.jsl の内容を確認します。

   package ステートメントがプロジェクト名と一致していることを確認します (たとえば、`package MyPackage`)。
3. ソリューション エクスプローラで、プロジェクト名を右クリックし、[新しい項目の追加] をクリックします。
4. [クラス] をクリックし、名前として「**AnotherClass**」と入力します。
5. [追加] をクリックします。
6. AnotherClass.jsl の内容を確認します。

   Visual J# によって、package ステートメントが jsl. ファイル内に自動的に追加されていることを確認します。追加する各クラスは、同じパッケージ名を含みます(たとえば、`package MyPackage`)。

既存のクラスをパッケージにグループ化するには

1. 各 .jsl ファイルを開き、同じ package ステートメントを追加し、続けてセミコロン (;) を入力します。

   package ステートメントは、ソース ファイルの最初の行に配置する必要があります。この行をコメント行や空白行にすることはできません。
   例：`package mycompany.platform;`
2. F5 キーを押してアプリケーションをコンパイルおよび実行します。

   正常にコンパイルされると、クラスが 1 つの名前空間にグループ化され、対応するディレクトリ構造がセットアップされます。

参照

処理手順

Scoping サンプル (/securescoping の使用)

関連項目

/securescoping (パッケージ スコープを持つメンバへのアセンブリ外からのアクセスの禁止)

その他の技術情報

Visual J# コンパイラ

Visual J# のチュートリアル

以下の各チュートリアルでは、Visual J# アプリケーションを作成するときの Visual Studio 機能の使用方法について具体的に説明します。

XML Web サービスの作成とアクセスに関するチュートリアル

Visual J# による XML Web サービスの作成

ASP.NET Web サービス プロジェクトのテンプレートを使って単純な XML Web サービスを作成する手順について説明します。

Visual J# Web フォーム クライアントによる XML Web サービスへのアクセス

XML Web サービスの Web フォーム クライアントを作成する手順について説明します。

Visual J# Windows クライアントによる XML Web サービスへのアクセス

XML Web サービスの Windows フォーム クライアントを作成する手順について説明します。

コンポーネントおよびコントロール作成のチュートリアル

Visual J# によるコンポーネントの作成

簡単なコンポーネントの開発方法を示し、クライアントとコンポーネントの間のやり取り、オブジェクトの有効期間と循環参照、クライアントとコンポーネントのデバッグ、および共有メソッドとインスタンス メソッドの使い方について説明します。

Visual J# での簡単なマルチスレッド コンポーネントの作成

マルチスレッド コンポーネントの作成例を示し、スレッドのしくみ、およびコンポーネント内で複数のスレッドを調整する方法について説明します。

Visual J# でのユーザー コントロールの作成

UserControl クラスを継承する簡単なユーザー コントロールの開発方法と、作成したユーザー コントロールを継承する方法を示します。

Visual J# での Windows フォーム コントロールの継承

継承によって単純なボタン コントロールを作成する方法を示します。このボタンは、標準 Windows フォーム ボタンから機能を継承し、カスタム メンバを公開します。

関連項目

Visual J#

Visual J# ドキュメントのさまざまなトピックへのリンクの一覧です。

Visual J# のサンプル

Visual J# を使って .NET Framework 用のアプリケーションを記述する方法を示すサンプル ソース コードです。

Visual Studio に関するチュートリアル

各種の Visual Studio プログラミング言語を使用してさまざまな種類のアプリケーションやコンポーネントを作成するための方法について段階的に説明します。

参照

その他の技術情報

Visual J# リファレンス

XML Web サービスのチュートリアル (Visual J#)

このチュートリアルでは、XML Web サービスの作成と XML Web サービスへのアクセスという、2 つの独立した開発工程について説明します。XML Web サービスへのアクセスのチュートリアルでは、Web フォーム クライアントを使う場合と Windows クライアントを使う場合の例を示します。

このセクションの内容

チュートリアル : Visual J# による XML Web サービスの作成

ASP.NET Web サービス プロジェクトのテンプレートを使って単純な XML Web サービスを作成する手順について説明します。

チュートリアル : Visual J# Web フォーム クライアントによる XML Web サービスへのアクセス

XML Web サービスの Web フォーム クライアントを作成する手順について説明します。

チュートリアル : Visual J# Windows クライアントによる XML Web サービスへのアクセス

XML Web サービスの Windows フォーム クライアントを作成する手順について説明します。

関連するセクション

Visual J# のチュートリアル

分散 Web アプリケーションの作成、Windows フォームと Web フォームの操作、および XML Web サービスの作成とアクセスの方法について説明します。

XML Web サービスの作成とアクセスに関するチュートリアル

Visual C#、Visual Basic、ATL Server、および C++ を使用して XML Web サービスを開発する方法を示します。また、XML Web サービスの使用に関するチュートリアルもあります。

マネージ コードを使用した XML Web サービス

Visual Basic または Visual C# を使用して XML Web サービスを作成、配布、および利用するための技術について説明します。

Visual J# リファレンス

J# キーワード、J# 演算子、J# クラスなどの参照ドキュメントへのリンクを示します。

チュートリアル : Visual J# による XML Web サービスの作成

このチュートリアルでは、Visual J# を使って単純な XML Web サービスを作成する手順について説明します。

XML Web サービスを使うと、リモートのサーバー機能を利用できます。XML Web サービスによって、企業のデータやビジネス ロジックに対するプログラム用インターフェイスが公開できるようになります。つまり、クライアント アプリケーションとサーバー アプリケーションでそれらを取得し、操作できます。XML Web サービスでは、ファイアウォールを越えてデータを転送する場合に、HTTP や XML メッセージングなどの標準を使って、クライアントとサーバー間またはサーバーとサーバー間でデータを交換できます。

XML Web サービスの詳細については、「Acronym-WebService サンプル (XML Web サービスの作成と利用)」を参照してください。

メモ :

使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

Visual J# を使って XML Web サービスを作成するには

1. [ファイル] メニューの [新規作成] をポイントし、[Web サイト] をクリックします。

   [新しい Web サイト] ダイアログ ボックスが表示されます。

2. [ASP.NET Web サービス] アイコンをクリックします。

3. [言語] ボックスで、[Visual J#] を選択します。

4. [場所] ボックスで、[HTTP] を選択します (ボックスに表示されていない場合)。

5. [場所] ボックスに、Web サービスの名前と場所を入力します。アプリケーション名として、「**VJSharpWebService**」を指定します。

   既定では、プロジェクトにローカル コンピュータが使用されます。Web サービスのアドレスとして "http://localhost" が使用されます。

6. [OK] をクリックします。

   Visual Studio テンプレートによってソリューションが作成されます。Web サービス ファイル Service.asmx と、それに対応する分離コード ファイル Service.jsl が作成されます。

7. ソリューション エクスプローラで、Service.asmx を右クリックし、[コードの表示] をクリックします。

   コード エディタで、分離コード ファイル Service.jsl が開きます。

   コード エディタで、`HelloWorld` メソッドの次のコードに注意してください。

   その結果、メソッドは次のようになります。

   ```
   /** @attribute WebMethod() */
   public String HelloWorld()
   {
      return "Hello World";
   }
   ```

   `WebMethod` 属性に注意してください。この属性は、`HelloWorld` メソッドが Web サービス メソッドとして公開されることを示しています。

8. [デバッグ] メニューの [デバッグなしで開始] をクリックしてプロジェクトを実行します。

   プロジェクトがビルドされ、実行されます。Internet Explorer で、URL http://localhost/VJSharpWebService/Service.asmx が開きます。これは、Web サービスのすべての Web サービス メソッドのテスト ページです。

9. [HelloWorld] ハイパーリンクをクリックします。

   Internet Explorer で、http://localhost/VJSharpWebService/Service.asmx?op=HelloWorld が開きます。これは、HelloWorld Web サービス メソッドのテスト ページです。

10. [起動] をクリックします。

Internet Explorer で新しいページが開き、HelloWorld Web サービス メソッドが生成した次のような XML コードが表示されます。

```
<?xml version="1.0" encoding="utf-8" ?>
   <string xmlns="http://tempuri.org/">Hello World</string>
```

この XML Web サービスにアクセスするクライアントを作成するには、次のどちらかのトピックを参照してください。

- チュートリアル : Visual J# Web フォーム クライアントによる XML Web サービスへのアクセス
- チュートリアル : Visual J# Windows クライアントによる XML Web サービスへのアクセス

参照

その他の技術情報

XML Web サービスのチュートリアル (Visual J#)

チュートリアル : Visual J# Web フォーム クライアントによる XML Web サービスへのアクセス

このチュートリアルでは、Visual J# を使って単純な Web フォーム アプリケーションを作成する手順について説明します。このアプリケーションは、「チュートリアル : Visual J# による XML Web サービスの作成」で作成される XML Web サービスのクライアントです。このチュートリアルは、XML Web サービスのチュートリアルが既に実行済みであることを前提としています。

ASP.NET のコンポーネントである Web フォームを使うと、任意のマークアップ言語で Web ページのフォームを作成し、任意のブラウザに情報を表示できます。また、サーバーのコードを使ってアプリケーション ロジックを実装することもできます。詳細については、「ASP.NET Web ページ (Visual Studio)」を参照してください。

このチュートリアルでは、最初に Web フォーム クライアントを作成します。

Web フォーム クライアントの作成

Visual J# を使って Web フォーム クライアントを作成するには

1. [ファイル] メニューの [新規作成] をポイントし、[Web サイト] をクリックします。

   [新しい Web サイト] ダイアログ ボックスが表示されます。

2. [ASP.NET Web アプリケーション] アイコンをクリックします。

3. [言語] ボックスの一覧の Visual J# をクリックします。

4. [場所] ボックスで、[HTTP] を選択します (ボックスに表示されていない場合)。

5. [場所] ボックスに、Web アプリケーションの名前と場所を入力します。アプリケーション名として、「**VJSharpWebForms**」を指定します。既定では、プロジェクトにローカル コンピュータが使用されます。Web アプリケーションのアドレスとして "http://localhost" が使用されます。

6. [OK] をクリックします。

   Visual Studio テンプレートによってソリューションが作成されます。プロジェクト テンプレートにより、テンプレート ファイル (default.aspx) がプロジェクトに追加され、このファイルがデザイン ビューで開かれます。既定のテンプレートには、フォーム コントロールが含まれています。また、プロジェクト テンプレートにより分離コード ファイル Web.Config が作成されます。

7. ツールボックスを開き、Label コントロールと Button コントロールをデザイン サーフェイスにドラッグします。

   Label1 コントロールと **Button1** コントロールが作成されます。

8. ボタンのテキスト プロパティを "Click Here" に変更します。

9. ボタンをダブルクリックします。

   `Button1_Click` イベント ハンドラが作成され、コード エディタが開きます。

10. イベント ハンドラ本体に次のコードを挿入します。

```
Label1.set_Text ("Hello World");
```

11. [デバッグ] メニューの [デバッグなしで開始] をクリックしてプロジェクトを実行します。

    プロジェクトがビルドされ、次の URL を持つ Web ページが作成されて、Internet Explorer に表示されます。

```
http://localhost/VJSharpWebForms/default.aspx
```

12. フォームの [Click Here] ボタンをクリックします。

    ラベルのテキストが "Hello World" に変更されます。

XML Web サービスの利用

以下の手順は、「チュートリアル : Visual J# による XML Web サービスの作成」で XML Web サービスが作成されたことを前提としています。

XML Web サービスを利用するには

1. ソリューション エクスプローラで、VJSharpWebForms プロジェクトの [参照設定] ノードを右クリックし、[Web 参照の追加] をクリックします。

   [Web 参照の追加] ダイアログ ボックスが表示されます。

2. [URL] ボックスに次の Web サービス アドレスを入力し、[移動] をクリックします。

   ```
   http://localhost/VJSharpWebService/Service.asmx
   ```

   これは、「チュートリアル : Visual J# による XML Web サービスの作成」で作成した Web サービスの URI です。HelloWorld という Web サービス メソッドが、[Web 参照の追加] ダイアログ ボックスの左ペインに表示されます。

   メモ :

   [URL] ボックスに Web サービス アドレスを入力する代わりに、[Web 参照の追加] ダイアログ ボックスの [ローカル コンピュータの Web サービス] オプションをオンにすることによって特定の Web サービスを参照するように指定することもできます。

3. [参照の追加] をクリックします。

   VJSharpWebService Web サービスのプロキシ クラスが生成され、プロジェクトに追加されます。これで、ローカルに定義されているメソッドと同じように、Web サービス メソッドを呼び出すことができます。プロキシ クラスは、SOAP over HTTP トランスポートを使って、Web サービス メソッドの呼び出しを Web サービス サーバーにルーティングします。

4. ファイル default.aspx をクリックし、[デザイナの表示] をクリックしてデザイン サーフェイスに戻ります。

5. ツールボックスを開いて [標準] タブをクリックし、新しいラベルをデザイン サーフェイスにドラッグします。

   Label2 コントロールが作成されます。

6. [Click Here] ボタンをダブルクリックします。

   コード エディタが開きます。

7. イベント ハンドラ `Button1_Click` の本体に、以下の行を追加します。

   ```
   // Visual J#
   localhost.Service service = new localhost.Service();
   Label2.set_Text(service.HelloWorld());
   ```

8. F5 キーを押してプロジェクトを実行します。

   プロジェクトがビルドされ、作成された Web ページが Internet Explorer に表示されます。

9. フォームの [Click Here] ボタンをクリックします。

   両方のラベルに "Hello World" というテキストが表示されます。

参照

処理手順

チュートリアル : Visual J# Windows クライアントによる XML Web サービスへのアクセス

その他の技術情報

XML Web サービスのチュートリアル (Visual J#)

チュートリアル : Visual J# Windows クライアントによる XML Web サービスへのアクセス

このチュートリアルでは、Visual J# を使って単純な Windows フォーム アプリケーションを作成する手順について説明します。このアプリケーションは、「チュートリアル : Visual J# による XML Web サービスの作成」で作成される XML Web サービスのクライアントです。このチュートリアルは、XML Web サービスのチュートリアルが既に実行済みであることを前提としています。

最初に、クライアントを作成し、Web 参照を XML Web サービスに追加します。次に、クライアントのユーザー インターフェイスを変更して、クライアントが XML Web サービスを呼び出すことができるようにします。その後、クライアントをビルドし、テストします。

Windows アプリケーションの作成

XML Web サービス クライアントにアクセスする Windows アプリケーションを作成するには

1. [ファイル] メニューの [開く] をポイントし、[プロジェクト/ソリューション] をクリックします。

   [ソリューションを開く] ダイアログ ボックスが表示されます。

2. VJSharpWebService ソリューションを選択し、[開く] をクリックします。

3. ソリューション エクスプローラで、ソリューション ルート VJSharpWebService を右クリックし、[追加] をポイントして、[新しいプロジェクト] をクリックします。

   [新しいプロジェクトの追加] ダイアログ ボックスが表示されます。

4. [プロジェクトの種類] ボックスの一覧の [Visual J#] をクリックし、[Windows アプリケーション] アイコンをクリックします。

5. [プロジェクト名] ボックスに「VJSharpWinClient」と入力します。

6. [OK] をクリックします。

   プロジェクト テンプレートによって VJSharpWinClient プロジェクトが作成され、Windows フォーム デザイナが表示されます。

7. ソリューション エクスプローラで、プロジェクト ノード VJSharpWinClient を右クリックし、[スタートアップ プロジェクトに設定] をクリックします。

8. ソリューション エクスプローラで、VJSharpWinClient プロジェクトの [参照設定] ノードを右クリックし、[Web 参照の追加] をクリックします。

   [Web 参照の追加] ダイアログ ボックスが開きます。

9. [URL] ボックスに次の Web サービス アドレスを入力し、[移動] をクリックします。

   ```
   http://localhost/VJSharpWebService/Service.asmx
   ```

   これは、「チュートリアル : Visual J# による XML Web サービスの作成」で作成した Web サービスの URI です。`HelloWorld` という Web サービス メソッドが、[Web 参照の追加] ダイアログ ボックスの左ペインに表示されます。

   メモ :

   [URL] ボックスに Web サービス アドレスを入力する代わりに、[Web 参照の追加] ダイアログ ボックスの [ローカル コンピュータの Web サービス] オプションをオンにすることによって特定の Web サービスを参照するように指定することもできます。

10. [参照の追加] をクリックします。

    VJSharpWebService Web サービスのプロキシ クラスが生成され、プロジェクトに追加されます。これで、ローカルに定義されているメソッドと同じように、Web サービス メソッドを呼び出すことができます。プロキシ クラスは、SOAP over HTTP トランスポートを使って、Web サービス メソッドの呼び出しを Web サービス サーバーにルーティングします。

XML Web サービスを呼び出すためのクライアントの変更

次に、クライアントのユーザー インターフェイスを変更して、クライアントが XML Web サービスを呼び出すことができるようにします。その後、クライアントをビルドし、テストします。

クライアントのユーザー インターフェイスを変更するには

1. ツールボックスを開き、ラベル コントロールとボタン コントロールを Form1 にドラッグします。

   label1 コントロールと **button1** コントロールが作成されます。

2. button1 のテキスト プロパティを "Click Here" に変更します。

3. [Click Here] ボタンをダブルクリックします。

   Form1.jsl ソース ファイルに、`button1_Click` イベント ハンドラが作成されます。コード エディタが開きます。

4. イベント ハンドラの本体を次のコードに置き換えます。

```
// Visual J#
VJSharpWinClient.localhost.Service service = new VJSharpWinClient.localhost.Service();
label1.set_Text(service.HelloWorld());
```

5. F5 キーを押してプロジェクトを実行します。

   プロジェクトがビルドされ、実行されます。

6. フォームで、button1 をクリックします。

   label1 のテキストが `"Hello World"` に変わります。

参照

処理手順

チュートリアル : Visual J# Web フォーム クライアントによる XML Web サービスへのアクセス

その他の技術情報

XML Web サービスのチュートリアル (Visual J#)

コンポーネントおよびコントロール作成のチュートリアル

以下のチュートリアルでは、コンポーネント、コントロール、およびスレッド処理を使用する方法について説明します。

このセクションの内容

チュートリアル : Visual J# によるコンポーネントの作成

簡単なコンポーネントの開発方法を示し、クライアントとコンポーネントの間のやり取り、オブジェクトの有効期間と循環参照、クライアントとコンポーネントのデバッグ、および共有メソッドとインスタンス メソッドの使い方について説明します。

チュートリアル : Visual J# での簡単なマルチスレッド コンポーネントの作成

マルチスレッド コンポーネントの作成例を示し、スレッドのしくみ、およびコンポーネント内で複数のスレッドを調整する方法について説明します。

チュートリアル : Visual J# でのユーザー コントロールの作成

UserControl クラスを継承する簡単なユーザー コントロールの開発方法と、作成したユーザー コントロールを継承する方法を示します。

チュートリアル : Visual J# での Windows フォーム コントロールの継承

継承によって単純なボタン コントロールを作成する方法を示します。このボタンは、標準 Windows フォーム ボタンから機能を継承し、カスタム メンバを公開します。

関連するセクション

Visual J# のチュートリアル

分散 Web アプリケーションの作成、Windows フォームと Web フォームの操作、および XML Web サービスの作成とアクセスの方法について説明します。

コンポーネントによるプログラミング

サーバーで使うコンポーネントを含めたすべてのコンポーネントについて説明する Visual Basic および Visual C# 関連のセクションです。

Visual J# リファレンス

J# キーワード、J# 演算子、J# クラスなどの参照ドキュメントへのリンクを示します。

チュートリアル : Visual J# によるコンポーネントの作成

コンポーネントは、再利用できるコードをオブジェクトの形式で提供します。オブジェクトを作成し、オブジェクトのプロパティやメソッドを呼び出すことによってコンポーネントのコードを使うアプリケーションは、クライアントと呼ばれます。クライアントは、使用するコンポーネントと同じアセンブリに含まれている場合もあれば、別の場所に存在する場合もあります。

ここで説明する作成手順は互いに関連しているため、各手順の実行順序が重要になります。

プロジェクトの作成

CDemoLib クラス ライブラリと CDemo コンポーネントを作成するには

1. [ファイル] メニューの [新規作成] をクリックし、[プロジェクト] をクリックして [新しいプロジェクト] ダイアログ ボックスを開きます。Visual J# プロジェクトの一覧の [クラス ライブラリ] プロジェクト テンプレートを選択し、[プロジェクト名] ボックスに「CDemoLib」と入力します。

   メモ :

   新しいプロジェクトを作成するときには、必ずプロジェクトの名前を指定します。これにより、ルート パッケージ、アセンブリ名、およびプロジェクト名が設定されます。また、既定のコンポーネントも適切なパッケージに含まれるようになります。

2. [OK] をクリックします。
3. [プロジェクト] メニューの [コンポーネントの追加] をクリックします。
4. [新しい項目の追加] ダイアログ ボックスの [コンポーネント クラス] を選択し、[ファイル名] ボックスに「**CDemo.jsl**」と入力します。[追加] をクリックします。

   CDemo.jsl という名前のコンポーネントがクラス ライブラリに追加されます。

5. ソリューション エクスプローラで CDemo.jsl を右クリックし、[コードの表示] をクリックします。コード エディタが開きます。

   `public class CDemo` のすぐ後ろに、`extends System.ComponentModel.Component` というコードが追加されていることを確認します。この部分は、`CDemo` クラスの継承元のクラスを示します。既定では、コンポーネントはシステムによって提供される Component クラスを継承します。Component クラスには、デザイナを使う機能を含め、コンポーネントのための多くの機能が用意されています。

6. ソリューション エクスプローラで、[Class1.jsl] を右クリックし、[削除] をクリックします。

   これにより、クラス ライブラリで提供される既定のクラスが削除されます。このクラスは、このチュートリアルでは使用しません。

7. [ファイル] メニューの [すべてを保存] をクリックして、プロジェクトを保存します。[プロジェクトの保存] ダイアログ ボックスが表示されます。このダイアログ ボックスには、[プロジェクト名]、[場所]、および [新しいソリューション名] の各項目が表示されます。これらの項目は、プロジェクトを保存するときに変更できます。
8. [保存] をクリックします。

コンストラクタと Finalize メソッドの追加

コンストラクタは、コンポーネントの初期化方法を制御します。Finalize メソッドは、コンポーネントの終了方法を制御します。CDemo クラスのコンストラクタと Finalize メソッドのコードには、存在する CDemo オブジェクトの現在の個数が保持されています。

CDemo クラスのコンストラクタと Finalize メソッドのコードを追加するには

1. コード エディタで、CDemo クラスのインスタンスの現在の個数と、各インスタンスの ID 番号を保持するメンバ変数を CDemo クラスに追加します。

```
// Visual J#
private long InstanceID;
private static long NextInstanceID = 0;
private static long ClassInstanceCount = 0;
```

   `ClassInstanceCount` メンバ変数と `NextInstanceID` メンバ変数は **static** と宣言されるため、これらの変数はクラス レベルでだけ存在します。これらのメンバにアクセスする `CDemo` のインスタンスはすべて、メモリ上の同じ場所を使用します。共有メモリは、コード内で `CDemo` クラスが最初に参照されたときに初期化されます。これは、`CDemo` オブジェクトが最初に作成されたときや、クラスの共有メンバの 1

つが最初にアクセスされたときです。

2. `public CDemo()` コンストラクタと `public CDemo(System.ComponentModel.IContainer container)` コンストラクタを探します。

   これらは、`CDemo` クラスの既定のコンストラクタです。コンポーネントにはパラメータの異なるいくつかのコンストラクタを定義できますが、どのコンストラクタにもクラスと同じ名前を付ける必要があります。

メモ：
クラスのインスタンスを作成できるクライアントは、クラスのコンストラクタのアクセス レベルによって決まります。

3. 新しい `CDemo` が作成されたときにインスタンス数をインクリメントし、インスタンス ID 番号を設定するために、`public CDemo()` に次のコードを追加します。

メモ：
コードは必ず `InitializeComponent` の呼び出しの後に追加してください。この呼び出しの時点で、含まれるコンポーネントがすべて初期化されます。

   ```
   // Visual J#
   InstanceID = NextInstanceID ++;
   ClassInstanceCount ++;
   ```

メモ：
マルチスレッドに詳しいユーザーであれば、`InstanceID` の割り当てと `NextInstanceID` のインクリメントが分割できない操作であることがわかります。

4. コンストラクタの末尾の後に次のメソッドを追加します。

   ```
   // Visual J#
   protected void Finalize()
   {
       ClassInstanceCount --;
   }
   ```

   メモリ マネージャでは、CDemo オブジェクトによって占有されていたメモリを最後にクリアする直前に、Finalize を呼び出します。Finalize メソッドは、.NET Framework のすべての参照型のルートである Object に定義されています。Finalize をオーバーライドすることにより、コンポーネントがメモリから削除される直前にクリーンアップを実行できます。しかし、後でわかるように、あらかじめリソースを解放しておく理由はいくつもあります。

クラスへのメソッドの追加

CDemo クラスに含まれているメンバを使うと、任意の時点でメモリ内に存在する CDemo オブジェクトの数をクライアントで取得できます。

CDemo のインスタンスの数を取得するメソッドを作成するには

- CDemo クラスに次のメソッド宣言を追加して、クライアントが CDemo のインスタンスの数を取得できるようにします。

  ```
  // Visual J#
  public static long getInstanceCount()
  {
      return ClassInstanceCount;
  }
  ```

コンポーネントをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックします。

ビルドは、コンパイル エラーも警告もなしに成功する必要があります。

コンポーネントのテスト

コンポーネントをテストするには、そのコンポーネントを使用するプロジェクトが必要です。このプロジェクトは、[実行] ボタンを押したときに最初に起動するプロジェクトであることが必要です。

CDemoTest クライアント プロジェクトをソリューションのスタートアップ プロジェクトとして追加するには

1. [ファイル] メニューの [プロジェクトの追加] をポイントし、[新しいプロジェクト] をクリックします。[新しいプロジェクトの追加] ダイアログ ボックスが表示されます。
2. Visual J# プロジェクトの一覧の [Windows アプリケーション] プロジェクト テンプレートをクリックし、[プロジェクト名] ボックスに「**CDemoTest**」と入力します。次に、[OK] をクリックします。

   これにより、新しいプロジェクトがソリューションに追加され、デザイナ ビューで Form1.jsl が開きます。
3. ソリューション エクスプローラで CDemoTest を右クリックし、ショートカット メニューの [スタートアップ プロジェクトに設定] をクリックします。

CDemo コンポーネントを使うには、クライアント テスト プロジェクトにクラス ライブラリ プロジェクトへの参照を追加する必要があります。参照を追加した後で、テスト アプリケーションに **import** ステートメントを追加すると、コンポーネントの使用を単純化できます。

クラス ライブラリ プロジェクトへの参照を追加するには

1. ソリューション エクスプローラで、[CDemoTest] のすぐ下にある [参照設定] ノードを右クリックし、ショートカット メニューの [参照の追加] をクリックします。
2. [参照の追加] ダイアログ ボックスの [プロジェクト] タブをクリックします。
3. [CDemoLib] クラス ライブラリ プロジェクトをダブルクリックして、[選択されたコンポーネント] ボックスの一覧に追加します。CDemoTest プロジェクトの [参照設定] ノードの下に CDemoLib が表示されます。
4. ソリューション エクスプローラで Form1.jsl を右クリックし、ショートカット メニューの [コードの表示] をクリックします。

CDemoLib への参照を追加することにより、コード内で CDemo コンポーネントの完全修飾名 `CDemoLib.CDemo` を使用できます。

import ステートメントを追加するには

- Form1 のコード エディタの一番上にある **import** ステートメントの一覧に、次の **import** ステートメントを追加します。

  ```
  // Visual J#
  import CDemoLib.*;
  ```

  import ステートメントを追加することにより、ライブラリ名を省略でき、コンポーネントを CDemo として参照できます。

  これで、コンポーネントをテストするためのテスト プログラムを作成して使用できます。

オブジェクトの有効期間

CDemoTest プログラムでは、多数の CDemo オブジェクトの作成と解放によって、.NET Framework におけるオブジェクトの有効期間を示しています。

CDemo オブジェクトを作成および解放するコードを追加するには

1. Form1.jsl を右クリックし、[デザイナの表示] をクリックしてデザイン ビューを開きます。
2. ツールボックスの [すべての Windows フォーム] タブから、ボタンとタイマを Form1 のデザイン サーフェイスにドラッグします。

   timer1 コンポーネント アイコンが、フォームの下のデザイン サーフェイスに表示されます。
3. timer1 のアイコンをダブルクリックして、timer1 コンポーネントの Tick イベントに対するイベント処理メソッドを作成します。イベント処理メソッドに次のコードを追加します。

   ```
   // Visual J#
   this.set_Text("CDemo instances: " + CDemo.getInstanceCount());
   ```

タイマが時を刻むごとに、フォームのキャプションには、現在の CDemo クラスのインスタンスの数が表示されます。クラス名は、共有 InstanceCount プロパティの修飾子として使用されています。共有メンバにアクセスするために CDemo のインスタンスを作成する必要はありません。

4. Form1 (`public Form1()`) のコンストラクタを探し、`InitializeComponent()` の呼び出しの後に次のコードを追加します。

```
// Visual J#
timer1.set_Enabled(true);
```

これにより、フォームが作成されるとすぐにタイマが起動されます。

5. [Form1.jsl [デザイン]] タブをクリックしてデザイナに戻ります。

6. Form1 上の Button をダブルクリックして、ボタンの Click イベントに対するイベント処理メソッドを作成します。イベント処理メソッドに次のコードを追加します。

```
// Visual J#
CDemo cd;
int ct;
for (ct = 0; ct < 1000; ct++)
{
    cd = new CDemo();
}
```

このコードは、見慣れないものかもしれません。CDemo の各インスタンスが作成されるたびに、前のインスタンスは解放されます。**for** ループが終了すると、CDemo のインスタンスは 1 つだけ残ります。イベント処理メソッドが終了すると、変数 cd がスコープ外になるため、そのインスタンスも解放されます。

CDemoTest プロジェクトと CDemo プロジェクトを実行してデバッグするには

1. F5 キーを押してソリューションを起動します。

   クライアント プロジェクトが起動し、Form1 が表示されます。フォームのキャプションが "CDemo instances: 0" となっていることを確認します。

2. ボタンをクリックします。フォームのキャプションが "CDemo instances: 1000" となります。

   CDemo のインスタンスは、ボタンの Click イベント処理プロシージャが終了するまでにすべて解放されています。簡単に言うと、メモリ マネージャがバックグラウンドでオブジェクトを終了させる優先順位が低いためです。優先順位は、システムがメモリ不足になった場合にだけ高くなります。このようなガベージ コレクション スキームを使うと、オブジェクトを高速に割り当てることができます。

3. ボタンをさらに数回クリックしてキャプションを確認します。ある時点で、インスタンスの数が突然少なくなります。これは、メモリ マネージャが一部のオブジェクトのメモリをクリアしたことを意味します。

   メモ :

   10 回を超えてクリックしても CDemo のインスタンス数が減らない場合は、メモリの使用量を増やすようにコードの調整が必要になる場合があります。フォームを閉じて開発環境に戻り、**for** ループの繰り返し回数を 10000 に増やしてから、もう一度プロジェクトを実行します。

4. 手順 3. を繰り返します。今度はインスタンス数がさらに減り、メモリ マネージャはより多くのオブジェクトを終了します。

   実際には、手順 3 を繰り返すたびに、メモリ マネージャにかかわりなく、より多くの CDemo オブジェクトを割り当てられるようになります。これは、Visual Studio のより多くの部分がスワップ アウトされ、CDemo のインスタンス用に多くのメモリ領域が残されるためです。

5. フォームを閉じて開発環境に戻ります。

参照

その他の技術情報

コンポーネントによるプログラミング

コンポーネントおよびコントロール作成のチュートリアル

チュートリアル : Visual J# での簡単なマルチスレッド コンポーネントの作成

複数のタスクを同時に実行できるアプリケーションを作成できます。この機能は、マルチスレッドまたはフリー スレッドと呼ばれ、ユーザーからの入力を要求するプロセッサ集中型のコンポーネントを設計する強力な方法です。マルチスレッドを利用するコンポーネントの例としては、給与支払い情報を計算するコンポーネントがあります。プロセッサ集中型の給与計算を 1 つのスレッドで実行している間に、ユーザーがデータベースに入力したデータを別のスレッドで処理できます。プロセスを個別のスレッドで実行することによって、ユーザーはコンピュータの計算処理が完了するのを待たずに追加のデータを入力できます。このチュートリアルでは、複数の複雑な計算を同時に実行する簡単なマルチスレッド コンポーネントを作成します。

プロジェクトの作成

作成するアプリケーションは、1 つのフォームと 1 つのコンポーネントで構成されます。ユーザーが値を入力してコンポーネントに通知すると、計算が開始されます。フォームはコンポーネントから値を受け取り、その値をラベル コントロールに表示します。このコンポーネントはプロセッサ集中型の計算を実行し、完了時にフォームに通知します。

ユーザー インターフェイスから受け取った値を保持するために、コンポーネントにパブリック変数を作成します。また、これらの変数の値に基づいて計算を実行するためのメソッドをコンポーネントに実装します。

メモ :

一般に、値を計算する方法としては関数が適していますが、スレッド間で引数を渡したり値を返したりできません。スレッドに値を渡したり、スレッドから値を受け取ったりするための簡単な方法は多数あります。この例では、パブリック変数を更新して、ユーザー インターフェイスに値を返します。また、スレッドが実行を完了したときに、イベントを使って `main` メソッドに通知します。

メモ :

使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

フォームを作成するには

1. 新しい Windows アプリケーション プロジェクトを作成します。
2. アプリケーションに「**Calculations**」という名前を付け、Form1.jsl を「**frmCalculations.jsl**」に変更します。

   このフォームは作成するアプリケーションのプライマリ ユーザー インターフェイスとして機能します。
3. ソリューション エクスプローラで frmCalculations.jsl を右クリックし、[コードの表示] をクリックします。コード エディタが開きます。
4. [編集] メニューの [検索と置換] をクリックし、[フォルダを指定して置換] をクリックして、ソリューション全体で Form1 を frmCalculations.jsl に置き換えます。[すべて置換] をクリックします。
5. ソリューション エクスプローラで frmCalculations.jsl を右クリックし、[デザイナの表示] をクリックします。デザイナが表示されます。
6. 5 つの Label コントロール、4 つの Button コントロール、および 1 つの TextBox コントロールをフォームに追加します。
7. 各コントロールのプロパティを次のように設定します。

コントロール	名前	テキスト
Label1	**lblFactorial1**	(空白)
Label2	**lblFactorial2**	(空白)
Label3	**lblAddTwo**	(空白)
Label4	**lblRunLoops**	(空白)
Label5	**lblTotalCalculations**	(空白)
Button1	**btnFactorial1**	Factorial

Button2	**btnFactorial2**	Factorial - 1
Button3	**btnAddTwo**	Add Two
Button4	**btnRunLoops**	Run a Loop
Textbox1	**txtValue**	(空白)

Calculator コンポーネントを作成するには

1. [プロジェクト] メニューの [コンポーネントの追加] をクリックします。
2. コンポーネントに「**Calculator.jsl**」という名前を付けます。

Calculator コンポーネントにパブリック変数を追加するには

1. Calculator のコード エディタを開きます。
2. ステートメントを追加して、frmCalculations から各スレッドに値を渡すために使うパブリック変数を作成します。

   変数 `varTotalCalculations` はコンポーネントによって実行された現在までの計算処理の合計件数を保持し、その他の変数はフォームから値を受け取ります。

```
// Visual J#
public int varAddTwo;
public int varFact1;
public int varFact2;
public int varLoopValue;
public double varTotalCalculations = 0;
```

Calculator コンポーネントにメソッドとイベントを追加するには

1. コンポーネントがフォームに値を渡すために使用するイベントのデリゲートを宣言します。

   メモ：

   宣言するイベントは 4 つですが、2 つのイベントは同じシグネチャを持つため、作成する必要があるデリゲートは 3 つだけです。

   前の手順で作成した変数宣言の直後に、次のコードを追加します。

```
// Visual J#
// This delegate will be invoked with two of your events.
/**@delegate */
public delegate void FactorialCompleteHandler(double Factorial, double TotalCalculatio
ns);
/**@delegate */
public delegate void AddTwoCompleteHandler(int Result, double TotalCalculations);
/**@delegate */
public delegate void LoopCompleteHandler(double TotalCalculations, int Counter);
```

2. 前の手順のコードの直後に次のコードを追加して、アプリケーションとの通信にコンポーネントが使用するイベントを宣言します。

```
// Visual J#
public FactorialCompleteHandler  FactorialComplete1;
public FactorialCompleteHandler  FactorialMinusOneComplete1;
public AddTwoCompleteHandler  AddTwoComplete1;
public LoopCompleteHandler  LoopComplete1;
```

```
/** @event */
public void add_FactorialComplete(FactorialCompleteHandler e)
{
    FactorialComplete1 = (FactorialCompleteHandler) System.Delegate.Combine(this.Facto
rialComplete1, e);
}
/** @event */
public void remove_FactorialComplete(FactorialCompleteHandler e)
{
   FactorialComplete1 = (FactorialCompleteHandler) System.Delegate.Remove(this.Factori
alComplete1, e);
}

/** @event */
public void add_FactorialMinusOneComplete(FactorialCompleteHandler e)
{
   FactorialMinusOneComplete1 = (FactorialCompleteHandler) System.Delegate.Combine(thi
s.FactorialMinusOneComplete1, e);
}
/** @event */
public void remove_FactorialMinusOneComplete(FactorialCompleteHandler e)
{
   FactorialMinusOneComplete1 = (FactorialCompleteHandler) System.Delegate.Remove(this
.FactorialMinusOneComplete1, e);
}

/** @event */
public void add_AddTwoComplete(AddTwoCompleteHandler e)
{
   AddTwoComplete1 = (AddTwoCompleteHandler) System.Delegate.Combine(this.AddTwoComple
te1, e);
}
/** @event */
public void remove_AddTwoComplete(AddTwoCompleteHandler e)
{
   AddTwoComplete1 = (AddTwoCompleteHandler) System.Delegate.Remove(this.AddTwoComplet
e1, e);
}

/** @event */
public void add_LoopComplete(LoopCompleteHandler e)
{
   LoopComplete1 = (LoopCompleteHandler) System.Delegate.Combine(this.LoopComplete1, e
);
}
/** @event */
public void remove_LoopComplete(LoopCompleteHandler e)
{
   LoopComplete1 = (LoopCompleteHandler) System.Delegate.Remove(this.LoopComplete1, e)
;
}
```

3. 前の手順で追加したコードの直後に、次のコードを追加します。

```
// Visual J#
// This method will calculate the value of a number minus 1 factorial
// (varFact2-1!).
public void FactorialMinusOne()
{
```

```
    double varTotalAsOfNow = 0;
    double varResult = 1;
    // Performs a factorial calculation on varFact2 - 1.
    for (int varX = 1; varX <= varFact2 - 1; varX++)
    {
        varResult *= varX;
        // Increments varTotalCalculations and keeps track of the
        // current total as of this instant.

        varTotalCalculations += 1;
        varTotalAsOfNow = varTotalCalculations;

    }
    // Signals that the method has completed and communicates the
    // result and a value of total calculations performed up to this
    // point.
    FactorialMinusOneComplete1.Invoke(varResult, varTotalAsOfNow);
}

// This method will calculate the value of a number factorial.
// (varFact1!)
public void Factorial()
{
    double varResult = 1;
    double varTotalAsOfNow = 0;
    for (int varX = 1; varX <= varFact1; varX++)
    {
        varResult =varResult * varX;

        varTotalCalculations =varTotalCalculations + 1;
        varTotalAsOfNow = varTotalCalculations;

    }
    FactorialComplete1.Invoke(varResult, varTotalAsOfNow);
}

// This method will add 2 to a number (varAddTwo+2).
public void AddTwo()
{
    double varTotalAsOfNow = 0;
    int varResult = varAddTwo + 2;

    varTotalCalculations = varTotalCalculations + 1;
    varTotalAsOfNow = varTotalCalculations;

    AddTwoComplete1.Invoke(varResult, varTotalAsOfNow);
}

// This method will run a loop with a nested loop varLoopValue times.
public void RunALoop()
{
    int varX;
    double varTotalAsOfNow = 0;
    for (varX = 1; varX <= varLoopValue; varX++)
    {
        // This nested loop is added to slow down the program
        // and create a processor-intensive application.
        for (int varY = 1; varY <= 500; varY++)
        {
```

```
            varTotalCalculations += 1;
            varTotalAsOfNow = varTotalCalculations;
        }
    }
    LoopComplete1.Invoke(varTotalAsOfNow, varLoopValue);
}
```

コンポーネントへのユーザー入力の転送

次の手順では、ユーザーからの入力を受け取ったり、Calculator コンポーネントと値をやり取りしたりするためのコードを frmCalculations に追加します。

frmCalculations にフロントエンド機能を実装するには

1. コード エディタで frmCalculations.jsl を開きます。
2. `public class frmCalculations` ステートメントを探します。始まりの中かっこ ({) の直後に、次のコードを追加します。

```
// Visual J#
Calculator Calculator1;
```

3. コンストラクタを探します。`InitializeComponent()` の直後に、次の行を追加します。

```
// Visual J#
// Creates a new instance of Calculator.
Calculator1 = new Calculator();
```

4. デザイナで各ボタンをクリックして、各コントロールのクリック イベント ハンドラのコード アウトラインを生成します。ここに各ハンドラを作成するコードを追加します。

   完了すると、クリック イベント ハンドラは次のようになります。

```
// Visual J#
private void btnFactorial1_Click (Object sender, System.EventArgs e)
{
    Calculator1.varFact1 = Integer.parseInt(txtValue.get_Text());
    // Disables the btnFactorial1 until this calculation is complete.
    btnFactorial1.set_Enabled(false);
    Calculator1.Factorial();
}

private void btnFactorial2_Click (Object sender, System.EventArgs e)
{
    Calculator1.varFact2 = Integer.parseInt(txtValue.get_Text());
    btnFactorial2.set_Enabled(false);
    Calculator1.FactorialMinusOne();
}

private void btnAddTwo_Click (Object sender, System.EventArgs e)
{
    Calculator1.varAddTwo = Integer.parseInt(txtValue.get_Text());
    btnAddTwo.set_Enabled(false);
    Calculator1.AddTwo();
}

private void btnRunLoops_Click (Object sender, System.EventArgs e)
{
    Calculator1.varLoopValue = Integer.parseInt(txtValue.get_Text());
    btnRunLoops.set_Enabled(false);
```

```
    // Let the user know that a loop is running.
    lblRunLoops.set_Text( "Looping");
    Calculator1.RunALoop();
}
```

5. 前の手順で追加したコードの下に、フォームが Calculator1 から受け取るイベントを処理するコードを次のように追加します。

```
// Visual J#
protected void FactorialHandler(double Value, double Calculations)
// Displays the returned value in the appropriate label.
{
    lblFactorial1.set_Text(""+ Value);
    // Re-enables the button so it can be used again.
    btnFactorial1.set_Enabled( true);
    // Updates the label that displays the total calculations
    // performed.
    lblTotalCalculations.set_Text("TotalCalculations are " +
        Calculations);
}

protected void FactorialMinusHandler(double Value, double Calculations)
{
    lblFactorial2.set_Text(""+Value);
    btnFactorial2.set_Enabled(true);
    lblTotalCalculations.set_Text("TotalCalculations are " + Calculations);
}

protected void AddTwoHandler(int Value, double Calculations)
{
    lblAddTwo.set_Text(""+Value);
    btnAddTwo.set_Enabled(true);
    lblTotalCalculations.set_Text("TotalCalculations are " + Calculations);
}

protected void LoopDoneHandler(double Calculations, int Count)
{
    btnRunLoops.set_Enabled(true);
    lblRunLoops.set_Text(""+Count);
    lblTotalCalculations.set_Text("TotalCalculations are " + Calculations);
}
```

6. frmCalculations のコンストラクタの `Calculator1 = new Calculator();` 行の直後に、フォームが Calculator1 から受け取るカスタム イベントを処理するコードを追加します。

```
// Visual J#
Calculator1.add_FactorialComplete(new Calculator.FactorialCompleteHandler(FactorialHan
dler));
Calculator1.add_FactorialMinusOneComplete(new Calculator.FactorialCompleteHandler(this
.FactorialMinusHandler));
Calculator1.add_AddTwoComplete(new Calculator.AddTwoCompleteHandler(this.AddTwoHandler
));
Calculator1.add_LoopComplete(new Calculator.LoopCompleteHandler(this.LoopDoneHandler))
;
```

アプリケーションのテスト

これで、複数の複雑な計算を実行できるフォームとコンポーネントを取り込んだプロジェクトが作成されました。マルチスレッド機能はまだ実装されていませんが、先に進む前に、まずプロジェクトをテストして機能するかどうかを確認します。

プロジェクトをテストするには

1. F5 キーを押してソリューションを起動します。

   アプリケーションが開始され、frmCalculations が表示されます。

2. テキスト ボックスに「**4**」と入力し、[Add Two] をクリックします。

   下のラベルに数字 "6" が表示され、lblTotalCalculations には "Total Calculations are 1.0" と表示されます。

3. [Factorial - 1] をクリックします。

   ボタンの下に数字 "6.0" が表示され、lblTotalCalculations の表示が "Total Calculations are 4.0" に変わります。

4. テキスト ボックスの値を「**20**」に変更し、[Factorial] をクリックします。

   ボタンの下に数字 "2.43290200817664E18" が表示され、lblTotalCalculations の表示が "Total Calculations are 24.0" に変わります。

5. テキスト ボックスの値を「50000」に変更し、[Run a Loop] をクリックします。

   このボタンが再び使用できるようになるまでには、少し時間がかかります。このボタンの下のラベルに "50000" と表示され、計算処理の合計件数は "2.5000024E7" と表示されます。

6. テキスト ボックスの値を「500000」に変更して [Run a Loop] をクリックし、その直後に [Add Two] を 2 回クリックします。

   ボタンおよびフォームのすべてのコントロールは、ループ処理が完了するまで応答しません。プログラムを終了します。

   プログラムで実行されるスレッドが 1 つだけである場合は、上の例のようなプロセッサ集中型の計算を実行すると、計算が完了するまでプログラムが拘束される傾向があります。次のセクションでは、マルチスレッド機能をアプリケーションに追加して、複数のスレッドを同時に実行できるようにします。

マルチスレッド機能の追加

前の例ではシングル スレッドでだけ実行されるアプリケーションの制限について説明しました。次のセクションでは、Thread クラスのオブジェクトを使って、複数の実行スレッドをコンポーネントに追加します。

Threads メソッドを追加するには

1. コード エディタで Calculator.jsl を開きます。

2. Calculator クラスに次のメンバ変数を追加します。

```
// Visual J#
public System.Threading.Thread FactorialThread;
public System.Threading.Thread FactorialMinusOneThread;
public System.Threading.Thread AddTwoThread;
public System.Threading.Thread LoopThread;
```

3. ファイルの最後にあるクラス宣言が終わる直前に、次のメソッドを追加します。

```
// Visual J#
public void ChooseThreads(int threadNumber)
{
    // Determines which thread to start based on the value
    // it receives.
    switch(threadNumber)
    {
        case 1:
            // Sets the thread using the AddressOf the method where
            // the thread will start.
            FactorialThread = new System.Threading.Thread(new System.Threading.ThreadS
tart(this.Factorial));
            // Starts the thread.
            FactorialThread.Start();
            break;
```

```
        case 2:
            FactorialMinusOneThread = new System.Threading.Thread(new System.Threading
.ThreadStart(this.FactorialMinusOne));
            FactorialMinusOneThread.Start();
            break;
        case 3:
            AddTwoThread = new System.Threading.Thread(new
            System.Threading.ThreadStart(this.AddTwo));
            AddTwoThread.Start();
            break;
        case 4:
            LoopThread = new System.Threading.Thread(new
            System.Threading.ThreadStart(this.RunALoop));
            LoopThread.Start();
            break;
    }
}
```

Thread オブジェクトをインスタンス化するときは、ThreadStart オブジェクトの形式の引数が必要です。**ThreadStart** オブジェクトは、スレッドを開始するメソッドのアドレスを指すデリゲートです。**ThreadStart** オブジェクトは、パラメータを受け取ることも値を渡すこともできないため、void (Visual J#) メソッドを示すことしかできません。実装した `ChooseThreads` メソッドが、呼び出し元のプログラムから値を受け取り、この値を使用することにより、開始する必要のある適切なスレッドを決定します。

frmCalculations に適切なコードを追加するには

1. コード エディタで frmCalculations.jsl ファイルを開き、`private void btnFactorial1_Click` を見つけます。
   a. 直接 `Calculator1.Factorial1` メソッドを呼び出す行を次のようにコメント アウトします。

```
// Visual J#
// Calculator1.Factorial()
```

   b. `Calculator1.ChooseThreads` メソッドを呼び出すため、次の行を追加します。

```
// Visual J#
// Passes the value 1 to Calculator1, thus directing it to
// start the Correct thread.
Calculator1.ChooseThreads(1);
```

2. 他の `button_click` メソッドにも同様の変更を加えます。

メモ :

Threads 引数の適切な値が含まれていることを確認してください。

完成したコードは次のようになります。

```
// Visual J#
private void btnFactorial1_Click (Object sender, System.EventArgs e)
{
    Calculator1.varFact1 = Integer.parseInt(txtValue.get_Text());
    // Disables the btnFactorial1 until this calculation
    // is complete.
    btnFactorial1.set_Enabled(false);
    // Calculator1.Factorial();
    Calculator1.ChooseThreads(1);
}
```

```
private void btnFactorial2_Click (Object sender, System.EventArgs e)
{
    Calculator1.varFact2 = Integer.parseInt(txtValue.get_Text());
    btnFactorial2.set_Enabled(false);
    // Calculator1.FactorialMinusOne();
    Calculator1.ChooseThreads(2);
}

private void btnAddTwo_Click (Object sender, System.EventArgs e)
{
    Calculator1.varAddTwo = Integer.parseInt(txtValue.get_Text());
    btnAddTwo.set_Enabled(false);
    // Calculator1.AddTwo();
    Calculator1.ChooseThreads(3);
}

private void btnRunLoops_Click (Object sender, System.EventArgs e)
{
    Calculator1.varLoopValue = Integer.parseInt(txtValue.get_Text());
    btnRunLoops.set_Enabled(false);
    // Let the user know that a loop is running.
    lblRunLoops.set_Text( "Looping");
    // Calculator1.RunALoop();
    Calculator1.ChooseThreads(4);
}
```

コントロールのマーシャリング呼び出し

ここでは、フォームの表示を円滑に更新できるようにします。実行のメイン スレッドは常にコントロールを所有しているため、従属スレッドからコントロールを呼び出すにはマーシャリング呼び出しが必要です。マーシャリングは、スレッド境界を越えて呼び出しを移動する方法で、リソースの観点から考えるとたいへんコストがかかります。必要なマーシャリングの数量を最小化し、呼び出しがスレッド セーフな方法で処理されることを保証するには、`Control.BeginInvoke` メソッドを使って実行のメイン スレッド上のメソッドを呼び出します。このような呼び出しは、コントロールを操作するメソッドを呼び出すときに必要です。詳細については、「方法 : スレッドからコントロールを操作する」を参照してください。

コントロールを呼び出すプロシージャを作成するには

1. frmCalculations のコード エディタを開きます。宣言セクションに次のコードを追加します。

```
// Visual J#
/**@delegate */
public delegate void FHandler(double Value, double Calculations);
/**@delegate */
public delegate void A2Handler(int Value, double Calculations);
/**@delegate */
public delegate void LDHandler(double Calculations, int Count);
```

   `Invoke` と `BeginInvoke` では、適切なメソッドへのデリゲートが引数として必要です。これらの行では、適切なメソッドを呼び出すために `BeginInvoke` によって使用されるデリゲートのシグネチャを宣言します。

2. 次のように、空のメソッドをコードに追加します。

```
// Visual J#
public void FactHandler(double Value, double Calculations)
{
}
public void Fact1Handler(double Value, double Calculations)
{
}
public void Add2Handler(int Value, double Calculations)
```

```
{
}
public void LDoneHandler(double Calculations, int Count)
{
}
```

3. [編集] メニューの [切り取り] と [貼り付け] を使って、FactorialHandler メソッドのすべてのコードを切り取り、FactHandler に貼り付けます。

4. FactorialMinusHandler から Fact1Handler に、AddTwoHandler から Add2Handler に、および LoopDoneHandler から LDoneHandler に、前の手順を繰り返します。

   終了すると、FactorialHandler、Factorial1Handler、AddTwoHandler、および LoopDoneHandler に残されたコードはなく、これらに含まれていたすべてのコードは適切な新しいメソッドに移動されています。

5. BeginInvoke メソッドを呼び出して、メソッドを非同期的に呼び出します。BeginInvoke は、フォーム (this) またはフォーム上の任意のコントロールから呼び出すことができます。

   完了すると、コードは次のようになります。

```
// Visual J#
protected void FactorialHandler(double Value, double Calculations)
// Displays the returned value in the appropriate label.
{
    this.BeginInvoke(new FHandler(FactHandler), new Object[] {(System.Double)Value, (S
ystem.Double)Calculations});
}

protected void FactorialMinusHandler(double Value, double Calculations)
{
    this.BeginInvoke(new FHandler(Fact1Handler), new Object [] {(System.Double)Value,
(System.Double)Calculations});
}

protected void AddTwoHandler(int Value, double Calculations)
{
    this.BeginInvoke(new A2Handler(Add2Handler), new Object[] {(System.Int32)Value, (S
ystem.Double) Calculations});
}

protected void LoopDoneHandler(double Calculations, int Count)
{
    this.BeginInvoke(new LDHandler(LDoneHandler), new Object[]{(System.Double)Calculat
ions, (System.Int32)Count});
}
```

   一見すると、イベント ハンドラは単純に次のメソッドを呼び出しているように見えます。実際には、イベント ハンドラは処理のメイン スレッド上のメソッドを呼び出しています。この方法によりスレッドの境界を越えた呼び出しを減らし、ロックアップの発生を抑えながらマルチスレッド アプリケーションを効率的に実行できます。詳細については、「方法 : スレッドからコントロールを操作する」を参照してください。

6. 作業内容を保存します。

7. [デバッグ] メニューの [開始] をクリックしてソリューションをテストします。

   a. テキスト ボックスに「**1000000**」と入力し、[Run a Loop] をクリックします。

      このボタンの下のラベルに "Looping" と表示されます。このループの実行にはかなり時間がかかります。完了が早すぎる場合は、数字の大きさを適宜変更してください。

   b. まだ選択可能な 3 つのボタンを連続してクリックします。すべてのボタンが入力に応答することがわかります。[Add Two] の下のラベルが最初に結果を表示します。しだいに、[Factorial] ボタンの下のラベルに結果が表示されます。1,000,000 の階乗から返される数値は大きすぎて倍精度の変数には入りきらないため、これらの結果は無限になります。最後に、さらに遅れて、[Run a Loop]

ボタンの下に結果が返されます。

以上の結果から、4 つの異なる計算のセットが 4 つの異なるスレッドで同時に実行されました。ユーザー インターフェイスは入力に応答し続け、各スレッドの完了後に結果が返されました。

スレッドの調整

マルチスレッド アプリケーションを扱い慣れている方なら、入力したコードに不備があることに気付く場合があります。Calculator.jsl の計算を実行する各メソッドのコード行を思い出してください。

```
// Visual J#
varTotalCalculations = varTotalCalculations + 1;
varTotalAsOfNow = varTotalCalculations;
```

これらの 2 行のコードでは、パブリック変数 `varTotalCalculations` をインクリメントし、ローカル変数 `varTotalAsOfNow` をこの値に設定します。次に、この値は frmCalculations に返され、ラベル コントロールに表示されます。1 つのスレッドだけが実行されている場合は、正しい値が返されます。しかし、複数のスレッドが実行されている場合は、正しい値が返されることは保証されません。各スレッドは、変数 `varTotalCalculations` をインクリメントできます。あるスレッドによってこの変数がインクリメントされた後、インクリメントされた値が `varTotalAsOfNow` にコピーされる前に、別のスレッドによってインクリメントされ、変数の値が変更されてしまう可能性があります。つまり、実際には、各スレッドが正しくない結果をレポートしている可能性があります。

キーワード **synchronized** を使ってスレッドを同期化すると、各スレッドから常に正しい結果が返されます。構文は次のとおりです。

```
// Visual J#
synchronized(AnObject)
{
   // Insert code that affects the object.
   // Insert more code that affects the object.
   // Insert more code that affects the object.
   // Release the lock.
}
```

synchronized ブロックが入力されると、指定された式の実行は、指定されたスレッドが対象のオブジェクトを排他ロックするまでブロックされます。前の例では、`AnObject` で実行がブロックされます。このようにして、実行が他のスレッドによって干渉されることなく 1 つのブロックとして続行されます。1 つの単位として実行されるステートメントのセットは、分割不可能であることになります。`}` を検出すると、式が解放され、スレッドは通常どおりに実行されます。

synchronized ブロックをアプリケーションに追加するには

1. コード エディタで Calculator.jsl を開きます。
2. 次のコードが記述されたすべての場所を探します。

   ```
   // Visual J#
   varTotalCalculations = varTotalCalculations + 1;
   varTotalAsOfNow = varTotalCalculations;
   ```

   `calculation` メソッドごとに 1 つずつ、このコードが全部で 4 つ見つかります。
3. それぞれの場所でこのコードを次のように変更します。

   ```
   // Visual J#
   synchronized(this)
   {
       varTotalCalculations = varTotalCalculations + 1;
       varTotalAsOfNow = varTotalCalculations;
   }
   ```
4. 作業内容を保存し、前の例と同様にテストします。

   プログラムのパフォーマンスにわずかな影響があることがわかります。これはコンポーネントに排他ロックがかけられるときにスレッドの実行が停止するために起こります。この方法によって、正確さは保証されますが、複数のスレッドによって得られるパフォーマンス上の利点が若干損

なわれます。スレッドのロックは、必要かどうかを慎重に検討して、不可欠な場合にだけ実装してください。

参照

処理手順

方法 : 複数の実行スレッドを調整する

その他の技術情報

コンポーネントによるプログラミング

コンポーネントのマルチスレッド

コンポーネントおよびコントロール作成のチュートリアル

チュートリアル : Visual J# でのユーザー コントロールの作成

ユーザー コントロールは、カスタム グラフィカル インターフェイスの作成と再利用のための手段を提供します。ユーザー コントロールは、基本的には表示できる形式を持つコンポーネントです。そのため、ユーザー コントロールには、1 つ以上の Windows フォーム コントロール、コンポーネント、またはコード ブロックが含まれることがあります。これらの要素を使って、ユーザー入力の検証、表示プロパティの変更、または作成に必要なその他のタスクを実行することができ、機能を拡張できます。ユーザー コントロールは、他のコントロールと同じ方法で Windows フォームに配置できます。このチュートリアルでは、最初に ctlClock という名前の簡単なユーザー コントロールを作成します。チュートリアルの 2 番目の部分では、継承を使って ctlClock の機能を拡張します。

プロジェクトの作成

新しいプロジェクトを作成するときは、プロジェクト名を指定することによって、ルート パッケージ、アセンブリ名、およびプロジェクト名が設定され、既定のコンポーネントが適切なパッケージに含まれるようになります。

ctlClockLib コントロール ライブラリおよび ctlClock コントロールを作成するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックして [新しいプロジェクト] ダイアログ ボックスを表示します。
2. [Visual J# プロジェクト] ボックスの一覧の [Windows コントロール ライブラリ] プロジェクト テンプレートをクリックし、[プロジェクト名] ボックスに「**ctlClockLib**」と入力します。

   既定では、ctlClockLib というプロジェクト名がルート パッケージにも割り当てられます。ルート パッケージは、アセンブリ内のコンポーネントの名前を限定するために使用されます。たとえば、`ctlClock` という名前のコンポーネントが 2 つのアセンブリに含まれる場合は、`ctlClockLib.ctlClock` という形で目的の `ctlClock` コンポーネントを指定できます。
3. ソリューション エクスプローラで UserControl1.jsl を右クリックし、ショートカット メニューの [コードの表示] をクリックします。
4. ソリューション エクスプローラで、UserControl1.jsl を右クリックし、[名前の変更] をクリックします。名前を UserControl1.jsl から **ctlClock.jsl** に変更します。
5. クラス定義とクラス コンストラクタを調べて、それぞれの名前が `public class UserControl1` から `public class ctlClock`、`public UserControl1()` から `public ctlClock()` に変更されていることを確認します。
6. [ファイル] メニューの [すべてを保存] をクリックして、プロジェクトを保存します。

ユーザー コントロールへの Windows コントロールとコンポーネントの追加

ビジュアル インターフェイスは、ユーザー コンポーネントの基本的な部分です。このビジュアル インターフェイスは、1 つ以上のコントロールをユーザー コントロール デザイナに追加することによって実装されます。次の例では、コントロールをユーザー コントロールに取り込み、コードを作成して機能を実装します。

ラベルとタイマをユーザー コントロールに追加するには

1. ソリューション エクスプローラで ctlClock.jsl を右クリックし、[デザイナの表示] をクリックします。
2. ツールボックスの [すべての Windows フォーム] タブをクリックし、[label] をダブルクリックします。

   label1 という名前のラベル コントロールが、ユーザー コントロール デザイナのコントロールに追加されます。
3. デザイナで [label1] をクリックします。[プロパティ] ウィンドウで、次のプロパティを設定します。

プロパティ	変更後の値
Name	lblDisplay
Text	(空白)
TextAlign	MiddleCenter
Font.Size	14

4. ツールボックスの [すべての Windows フォーム] をクリックし、[Timer] をダブルクリックします。

   タイマはコンポーネントであるため、実行時には表示できる形式を持ちません。そのため、タイマはユーザー コントロール デザイナ上のコント

ロールと共には表示されず、コンポーネント トレイに表示されます。

5. コンポーネント トレイで、[timer1] をクリックし、Interval プロパティを 1000 に設定し、Enabled プロパティを True に設定します。

   Interval プロパティは、タイマ コンポーネントが時を刻む頻度を制御します。timer1 が時を刻むたびに、timer1_Tick イベントのコードが実行されます。時を刻む間隔は、ミリ秒単位で表します。

6. コンポーネント トレイで、[timer1] をダブルクリックして、ctlClock の timer1_Tick イベントに移動します。

7. コードを次のように変更します。

```
// Visual J#
protected void timer1_Tick (Object sender, System.EventArgs e)
{
    lblDisplay.set_Text(System.DateTime.get_Now().ToLongTimeString());
}
```

   このコードは、現在の時刻を lblDisplay に表示します。timer1 の間隔は「1000」に設定されているため、このイベントは 1000 ミリ秒ごとに発生します。したがって、現在の時刻は 1 秒ごとに更新されます。

   既定では、[timer1] のダブルクリックによって追加された timer1_Tick メソッドは、**private** スコープを持ちます。上のコード セグメントで、このスコープを **protected** に変更したことに注意してください。

8. [ファイル] メニューの [すべてを保存] をクリックしてプロジェクトを保存します。

ユーザー コントロールへのカスタム プロパティの追加

この時点で、作成した時計コントロールには、それぞれ固有のプロパティのセットを持つ Label コントロールと Timer コンポーネントがカプセル化されています。各コントロールのプロパティは、コントロールの後続のユーザーからはアクセスできませんが、適切なコード ブロックを作成することによってカスタム プロパティを作成および公開できます。次のセクションでは、コントロールにカスタム プロパティを追加して、ユーザーが背景色とテキストの色を変更できるようにします。

ユーザー コントロールにカスタム メンバを追加するには

1. ソリューション エクスプローラで ctlClock.jsl を右クリックし、ショートカット メニューの [コードの表示] をクリックします。

   コード エディタが開きます。

2. `public class ctlClock` ステートメントを探します。始まりの中かっこ ({) の下に、次のように入力します。

```
// Visual J#
private Color colFColor;
private Color colBColor;
```

   これらのステートメントは、これから作成するカスタム プロパティの値を格納するためのプライベート変数を作成します。

3. 前の手順の変数宣言の下に、次のコードを追加します。

```
// Visual J#
/** @property */
public Color get_ClockBackColor()
{
    return colBColor;
}
/** @property */
public void set_ClockBackColor(Color value)
{
    colBColor = value;
    lblDisplay.set_BackColor(colBColor);
}
/** @property */
public Color get_ClockForeColor()
{
```

```
    return colFColor;
}
/** @property */
public void set_ClockForeColor(Color value)
{
    colFColor = value;
    lblDisplay.set_ForeColor(colFColor);
}
```

上のコードにより、このコントロールの後続のユーザーは、ClockForeColor と ClockBackColor の 2 つのカスタム プロパティを使用できるようになります。get_ ステートメントと set_ ステートメントは、カスタム プロパティの値を格納および取得し、適切な機能を実装するコードです。

4. [ファイル] メニューの [すべてを保存] をクリックしてプロジェクトを保存します。

コントロールのテスト

コントロールはスタンドアロン アプリケーションではないため、コンテナでホストする必要があります。コントロールをテストするには、コントロールを実行するテスト プロジェクトを指定する必要があります。ここでは、コントロールを作成し、Windows フォームでそのコントロールをテストします。

コントロールをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックします。

テスト プロジェクトを作成するには

1. [ファイル] メニューの [追加] をポイントし、[新しいプロジェクト] をクリックして [新しいプロジェクトの追加] ウィンドウを開きます。
2. [Windows アプリケーション] をクリックし、[プロジェクト名] ボックスに「Test」と入力します。
3. ソリューション エクスプローラで、Test プロジェクトの [参照設定] ノードを右クリックします。[参照の追加] をクリックして [参照の追加] ダイアログ ボックスを表示します。
4. [プロジェクト] タブをクリックします。[プロジェクト名] ボックスにユーザー コントロール プロジェクトの名前が表示されます。

   プロジェクトをダブルクリックします。これで ctlClockLib Components という新しいタブがツールボックスに追加されました。

参照を追加した後で、コントロールをフォームに配置できます。

コントロールをテストするには

1. デザイン ビューに Form1 を表示した状態で、ツールボックスの [ctlClockLib コンポーネント] をクリックし、[ctlClock] を表すコントロール アイコンが表示されるまで、下にスクロールします。
2. [ctlClock] アイコンをダブルクリックします。

   コントロールのコピーがフォームに追加されます。このコントロールには現在の時刻が表示され、毎秒ごとに更新されます。
3. アイコンを選択し、フォーム上にマウスを移動します。
4. フォーム上でマウスを移動する間、マウスの左ボタンを押したままにします。

   コントロールのコピーがもう 1 つフォーム上に作成されます。必要に応じて、タイマのコピーをいくつでもフォームに追加できます。
5. デザイナで ctlClock のインスタンスの 1 つをクリックします。

   このインスタンスのプロパティが [プロパティ] ウィンドウに表示されます。
6. [プロパティ] ウィンドウで ClockBackColor プロパティを選択して、カラー パレットを表示します。
7. 任意の色をクリックして選択します。

   コントロールの背景色が、選択した色に変更されます。
8. 同様なイベントのシーケンスを使用して、ClockForeColor プロパティが予想どおりに機能することを確認します。
9. F5 キーを押し、[ユーザー コントロール テスト コンテナ] ウィンドウでこのコントロールを実行します。UserControl TestContainer の詳細については、「方法 : UserControl の実行時の動作をテストする」を参照してください。

10. [ユーザー コントロール テスト コンテナ] を閉じるには、[閉じる] をクリックします。

このセクションでは、コンポーネントと Windows コントロールをコードと結合し、ユーザー コントロールの形式にパッケージ化してカスタム機能を提供する方法を説明しました。ユーザー コントロールのプロパティを公開し、完了後にコントロールをテストする方法を学びました。次のセクションでは、ctlClock をベースとして使って、継承ユーザー コントロールを作成する方法を学びます。

ユーザー コントロールからの継承

前のセクションでは、再利用可能なユーザー コントロールに Windows コントロール、コンポーネント、およびコードを組み込む方法を学びました。このユーザー コントロールをベースにして、他のコントロールを作成できます。基本クラスからクラスを派生するプロセスを継承と呼びます。このセクションでは、ctlAlarmClock という名前のユーザー コントロールを作成します。このコントロールは、親コントロールの ctlClock から派生します。親メソッドをオーバーライドし、新しいメソッドやプロパティを追加して、ctlClock の機能を拡張する方法を学びます。

継承コントロールの作成

継承コントロールの作成の最初の手順は、親コントロールからの派生です。これにより、親コントロールのプロパティ、メソッド、およびグラフィカル特性をすべて持つ新しいコントロールが作成されます。このコントロールに基づいて、新しい機能を追加したり機能を変更したりできます。

継承コントロールを作成するには

1. ソリューション エクスプローラで、[ctlClockLib] をクリックします。
2. [プロジェクト] メニューの [ユーザー コントロールの追加] をクリックします。

   [ユーザー コントロール] が選択された状態で [新しい項目の追加] ウィンドウが表示されます。
3. [ファイル名] ボックスに「ctlAlarmClock.jsl」と入力し、[追加] をクリックします。

   [継承ピッカー] ウィンドウが表示されます。
4. [コンポーネント名] の [ctlClock] をダブルクリックします。
5. ソリューション エクスプローラで、現在のプロジェクトを参照します。ctlAlarmClock.jsl ファイルがプロジェクトに追加されていることを確認します。

アラーム プロパティの追加

ユーザー コントロールにプロパティを追加する場合と同じ方法で、継承コントロールにプロパティを追加できます。ここでは、プロパティ宣言構文を使って、コントロールに 2 つのプロパティを追加します。

- AlarmTime は、アラームが鳴り出す日付と時刻の値を格納します。
- AlarmSet は、アラームが設定されているかどうかを示します。

ユーザー コントロールにプロパティを追加するには

1. ソリューション エクスプローラで [ctlAlarmClock] を右クリックし、[コードの表示] をクリックします。
2. `public class` ステートメントを探します。このコントロールが ctlClockLib.ctlClock を継承していることを確認します。`public class ctlAlarmClock extends ctlClockLib.ctlClock {` ステートメントの下に次のコードを追加します。

```
// Visual J#
private System.DateTime dteAlarmTime;
private boolean blnAlarmSet;
// These properties will be declared as public to allow future
// developers to access them.
/** @property*/
public System.DateTime get_AlarmTime()
{
    return dteAlarmTime;
}
/** @property*/
public void set_AlarmTime(System.DateTime value)
{
    dteAlarmTime = value;
}
/** @property*/
```

```
public boolean get_AlarmSet()
{
    return blnAlarmSet;
}
/** @property*/
public void set_AlarmSet(boolean value)
{
    blnAlarmSet = value;
}
```

コントロールのグラフィカル インターフェイスへの追加

継承したコントロールには、継承元のコントロールと同じビジュアル インターフェイスがあります。継承したコントロールには親コントロールと同じ内在コントロールがありますが、内在コントロールのプロパティは特に公開されない限り使用できません。任意のユーザー コントロールに追加する場合と同じ方法で、継承されたユーザー コントロールのグラフィカル インターフェイスに追加できます。引き続き、アラーム クロックのビジュアル インターフェイスに、アラームが鳴ったときに点滅するラベル コントロールを追加します。

ラベル コントロールを追加するには

1. ソリューション エクスプローラで [ctlAlarmClock] を右クリックし、ショートカット メニューの [デザイナの表示] をクリックします。

   ctlAlarmClock のデザイナがメイン ウィンドウに表示されます。

2. コントロールの表示部分をクリックし、[プロパティ] ウィンドウの内容を確認します。

   すべてのプロパティが表示されていますが、淡色表示されていることに注意してください。つまり、これらのプロパティは lblDisplay に対してネイティブであり、[プロパティ] ウィンドウでは変更もアクセスもできません。既定では、ユーザー コントロールに含まれるコントロールは private であり、そのプロパティにアクセスする方法はありません。

ヒント
コントロールの後続のユーザーが内部コントロールにアクセスできるようにするには、それらをパブリックまたはプロテクトとして宣言します。これにより、適切なコードを使用して、ユーザー コントロールに含まれるコントロールのプロパティを設定したり変更したりできるようになります。

3. ユーザー コントロールに Label コントロールを追加します。
4. マウスを使用して、ラベル コントロールを表示ボックスのすぐ下に移動します。[プロパティ] ウィンドウで、次のプロパティを設定します。

プロパティ	設定
Name	lblAlarm
Text	Alarm!
TextAlign	Middle Center
Visible	False

アラーム機能の追加

前のセクションでは、ユーザー コントロールのアラーム機能を使用できるようにするカスタム メンバおよびコントロールを追加しました。このセクションでは、現在の時刻とアラームの時刻を比較し、一致した場合は、アラームを点滅させるコードを追加します。ctlClock の timer1_Tick メソッドをオーバーライドして、ctlClock の固有の機能をすべて維持したまま、ctlAlarmClock の機能を拡張します。

ctlClock の timer1_Tick メソッドをオーバーライドするには

1. コード エディタで、`private boolean blnAlarmSet;` ステートメントを探します。このステートメントの直後に、次のステートメントを追加します。

```
// Visual J#
private boolean blnColorTicker;
```

2. クラス宣言が終わる直前に、次のコードを追加します。

```
// Visual J#
protected void timer1_Tick(Object sender, System.EventArgs e)
{
   // Calls the timer1_Tick method of ctlClock.
   super.timer1_Tick(sender, e);
   // Checks to see if the alarm is set.
   if (blnAlarmSet == false)
       return;
   else
   // If the date, hour, and minute of the alarm time are the same as
   // now, cause the display to flash.
   {
        if (dteAlarmTime.get_Date() == System.DateTime.get_Now().get_Date() &&
            dteAlarmTime.get_Hour() == System.DateTime.get_Now().get_Hour() &&
            dteAlarmTime.get_Minute() == System.DateTime.get_Now().get_Minute())
        {
            // Makes lblAlarm visible, and changes the backcolor based
            // on the value of blnColorTicker. The backcolor of the
            // label will flash once per tick of the clock.
            lblAlarm.set_Visible(true);
            if (blnColorTicker == false)
            {
                lblAlarm.set_BackColor( Color.get_Red());
                blnColorTicker = true;
            }
            else
            {
                lblAlarm.set_BackColor(Color.get_Blue());
                blnColorTicker = false;
            }
        }
        else
        {
            // Once the alarm has sounded for a minute, the label is
            // made invisible again.
            lblAlarm.set_Visible(false);
        }
    }
}
```

このコードを追加することで、いくつかのタスクが実行されます。これにより、基本コントロールから継承したメソッドの代わりに、このメソッドがコントロールで使用されます。このメソッドを呼び出した場合、`super.timer1_Tick` ステートメントが呼び出されて、元のコントロールに組み込まれていたすべての機能がこのコントロールで再生されます。次に、追加のコードが実行され、アラーム機能が組み込まれます。アラームが作動すると、点滅するラベル コントロールが表示されます。

アラーム クロック コントロールは、ほとんど完成しています。後は、アラームを止める手段を実装するだけです。この方法を実装するには、ボタンを追加し、btnAlarmOff_Click メソッドにコードを追加します。

アラームを止める手段を実装するには

1. ソリューション エクスプローラで ctlAlarmClock.jsl を右クリックし、[デザイナの表示] をクリックします。

   デザイナが表示されます。

2. コントロールにボタンを追加します。ボタンのプロパティを次のように設定します。

プロパティ	値

Name	btnAlarmOff
Text	Disable Alarm

3. デザイナで [Disable Alarm] をダブルクリックします。

   コード エディタが開き、`private void btnAlarmOff_Click` 行が表示されます。

4. このメソッドを次のように変更します。

```
// Visual J#
private void btnAlarmOff_Click (Object sender, System.EventArgs e)
{
    // Turns off the alarm.
    blnAlarmSet = false;
    // Hides the flashing label
    lblAlarm.set_Visible(false);
}
```

5. [ファイル] メニューの [すべてを保存] をクリックしてプロジェクトを保存します。

継承コントロールのテスト

標準ユーザー コントロールと同様に、継承ユーザー コントロールはスタンドアロン型ではないため、フォームまたは他のコンテナでホストする必要があります。ctlAlarmClock には他にも機能があるため、テストを実行するには追加のコードが必要です。ここでは、ctlAlarmClock の機能をテストするための簡単なプログラムを記述します。ctlAlarmClock の AlarmTime プロパティを設定および表示するコードを作成し、固有の機能をテストします。

コントロールをビルドしてテスト フォームに追加するには

1. ソリューション エクスプローラで、[ctlClockLib] をクリックします。[ビルド] メニューの [ctlClockLib のビルド] をクリックします。
2. ソリューションに新しい Windows アプリケーション プロジェクトを追加し、Test2 という名前を付けます。
3. ソリューション エクスプローラで、テスト プロジェクトの [参照設定] ノードを右クリックします。[参照の追加] をクリックして [参照の追加] ウィンドウを表示します。[プロジェクト] タブをクリックします。[プロジェクト名] の一覧に [ctlClockLib] が表示されます。表示された [ctlClockLib] をダブルクリックして、[ctlClockLib コンポーネント] ウィンドウに表示されていることを確認します。これで ctlClockLib への参照が追加されます。
4. ツールボックス の [ctlClockLib コンポーネント] をクリックします。
5. ctlAlarmClock のアイコンが表示されるまで、下方にスクロールします。
6. [ctlAlarmClock] をダブルクリックして、[ctlAlarmClock] をフォームにコピーします。
7. ツールボックスで [DateTimePicker] を探してダブルクリックし、DateTimePicker コントロールをフォームに追加します。また、[Label] をダブルクリックし、Label コントロールを追加します。
8. マウスを使用して、フォーム上で各コントロールを使いやすい位置に移動します。
9. 各コントロールのプロパティを次のように設定します。

コントロール	プロパティ	値
label1	Text	(空白のまま)
	Name	lblTest
dateTimePicker1	Name	dtpTest
	Format	Time

10. デザイナで [DateTimePicker] をダブルクリックします。

コード エディタが開き、`private void dtpTest_ValueChanged` が表示されます。

11. コードを次のように変更します。

```
// Visual J#
private void dtpTest_ValueChanged (Object sender, System.EventArgs e)
{
    ctlAlarmClock1.set_AlarmTime(dtpTest.get_Value());
    ctlAlarmClock1.set_AlarmSet(true);
    lblTest.set_Text("Alarm Time is " + ctlAlarmClock1.get_AlarmTime().ToShortTimeStri
ng());
}
```

12. ソリューション エクスプローラで [Test2] を右クリックし、ショートカット メニューの [スタートアップ プロジェクトに設定] をクリックします。

13. [デバッグ] メニューの [開始] をクリックします。

    テスト プログラムが起動します。ctlAlarmClock コントロールの現在時刻が更新され、DateTimePicker コントロールに開始時刻が表示されます。

14. DateTimePicker で分が表示されている部分をクリックします。

15. キーボードを使用して、ctlAlarmClock によって表示されている現在の時刻より 1 分後の値を設定します。

    アラーム設定時刻は lblTest に表示されます。

16. 表示時刻がアラームの設定時刻になるまで待ちます。

    表示時刻がアラームの設定時刻になったときに、lblAlarm が点滅します。[Disable Alarm] をクリックしてアラームを切ります。ここでアラームをリセットできます。

このチュートリアルでは、多数の重要な概念を扱いました。ユーザー コントロール コンテナにコントロールやコンポーネントを組み込んでユーザー コントロールを作成する方法を学びました。また、コントロールにプロパティを追加する方法や、カスタム機能を実装するコードを記述する方法も学びました。2 番目のセクションでは、継承によって特定のユーザー コントロールの機能を拡張し、オーバーライドによりホストのメソッドの機能を変更する方法を学びました。

参照

処理手順

チュートリアル : Visual J# での Windows フォーム コントロールの継承

その他の技術情報

コンポーネントによるプログラミング

コンポーネントおよびコントロール作成のチュートリアル

チュートリアル : Visual J# での Windows フォーム コントロールの継承

Visual J# には、継承を使って強力なカスタム コントロールを作成する機能が用意されています。継承を使用すると、標準の Windows フォーム コントロールの固有の機能をすべて保持しながら、カスタム機能も組み込んだコントロールを作成できます。このチュートリアルでは、ValueButton という名前の簡単な継承コントロールを作成します。このボタンは、標準 Windows フォーム ボタンの機能を継承し、ButtonValue という名前のカスタム メンバを公開します。

メモ :

使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

プロジェクトの作成

新しいプロジェクトを作成するときは、名前を指定することによって、ルート パッケージ、アセンブリ名、およびプロジェクト名が設定され、既定のコンポーネントが適切なパッケージに含まれるようになります。

ValueButtonLib コントロール ライブラリおよび ValueButton コントロールを作成するには

1. [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックして [新しいプロジェクト] ダイアログ ボックスを開きます。
2. [Visual J# プロジェクト] ボックスの一覧の [Windows コントロール ライブラリ] プロジェクト テンプレートをクリックし、[プロジェクト名] ボックスに「**ValueButtonLib**」と入力します。
3. [OK] をクリックします。

   既定では、プロジェクト名の ValueButtonLib がルート パッケージにも割り当てられます。ルート パッケージは、アセンブリ内のコンポーネントの名前を限定するために使用されます。たとえば、ValueButton という名前のコンポーネントが 2 つのアセンブリに含まれる場合は、`ValueButtonLib.ValueButton` という形で ValueButton を指定します。
4. ソリューション エクスプローラで、UserControl1.jsl を右クリックし、ショートカット メニューの [名前の変更] をクリックします。ファイル名を **ValueButton.jsl** に変更します。
5. ソリューション エクスプローラで [ValueButton.jsl] を右クリックし、[コードの表示] をクリックします。
6. `class` ステートメント、`public class ValueButton` を検索し、このコントロールが継承した型を `System.Windows.Forms.UserControl` から `System.Windows.Forms.Button` に変更します。これにより、継承したコントロールは、Button コントロールのすべての機能を継承できます。
7. [コンポーネント デザイナで生成されたコード] を展開し、`InitializeComponent` メソッドを検索し、AutoScaleMode プロパティを割り当てた行を削除します。

   ```
   this.set_AutoScaleMode(System.Windows.Forms.AutoScaleMode.Font);
   ```

   このプロパティは、Button コントロール内には存在しません。
8. [ファイル] メニューの [すべてを保存] をクリックして、プロジェクトを保存します。

   ビジュアル デザイナが使用できなくなります。**Button** コントロールは独自の描画を実行するため、デザイナでは表示を変更できません。ビジュアル表示は、コードで変更しない限り、継承元のクラス (**Button**) とまったく同じです。

   メモ :

   UI 要素のないコンポーネントをデザイン サーフェイスに追加することはできます。

継承コントロールへのメンバの追加

継承された Windows フォーム コントロールの使用方法の 1 つとして考えられるのは、標準 Windows フォーム コントロールと同じコントロールを作成して、カスタム プロパティを公開することです。このセクションでは、コントロールに ButtonValue という名前のメンバを追加します。

ButtonValue メンバを追加するには

1. ソリューション エクスプローラで ValueButton.jsl を右クリックし、ショートカット メニューの [コードの表示] をクリックします。
2. `class` ステートメントを探します。始まりの中かっこ ({) の直後に、次のコードを追加します。

```
// Visual J#
// Creates the private variable that will store the value of your
// member.
private int varValue;
// Declares the methods to access or modify the member.
/** @property */
public int get_ButtonValue()
{
    return varValue;
}
/** @property */
public void set_ButtonValue(int value)
{
    varValue = value;
}
```

   このコードは、`ButtonValue` メンバを格納および取得するメソッドを設定します。**get_** ステートメントは、返された値をプライベート変数 `varValue` に格納されている値に設定します。**set_** ステートメントは、**value** キーワードを使用してプライベート変数の値を設定します。

3. [ファイル] メニューの [すべてを保存] をクリックして、プロジェクトを保存します。

コントロールのテスト

コントロールはスタンドアロン プロジェクトではないため、コンテナでホストする必要があります。コントロールをテストするには、コントロールを実行するテスト プロジェクトを指定する必要があります。また、コントロールをビルドしてテスト プロジェクトからもアクセスできるようにする必要があります。ここでは、コントロールを作成し、Windows フォームでそのコントロールをテストします。

コントロールをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックします。

テスト プロジェクトを作成するには

1. [ファイル] メニューの [プロジェクトの追加] をポイントし、[新しいプロジェクト] をクリックして [新しいプロジェクトの追加] ダイアログ ボックスを開きます。
2. [Visual J# プロジェクト] ノードを選択し、[Windows アプリケーション] をクリックします。
3. [プロジェクト名] ボックスに「**Test**」と入力し、[OK] をクリックします。
4. ソリューション エクスプローラで、テスト プロジェクトの [参照設定] ノードを右クリックし、ショートカット メニューの [参照の追加] をクリックして [参照の追加] ダイアログ ボックスを表示します。
5. [プロジェクト] タブをクリックします。
6. ValueButtonLib プロジェクトをダブルクリックして、テスト プロジェクトに参照を追加します。

フォームにコントロールを追加するには

1. ソリューション エクスプローラで Form1.jsl を右クリックし、ショートカット メニューの [デザイナの表示] をクリックします。
2. ツールボックスで、[ValueButton] アイコンをダブルクリックします。

   ValueButton のインスタンスがフォームに表示されます。

3. ValueButton を右クリックし、ショートカット メニューの [プロパティ] をクリックします。

   このコントロールのプロパティは [プロパティ] ウィンドウで確認できます。これらのプロパティは、標準ボタンにより公開されるプロパティと同じです。

4. ButtonValue プロパティを 5 に設定します。

5. ツールボックスの [コモン コントロール] タブで、[Label] をダブルクリックしてフォームに Label コントロールを追加します。
6. ラベルをフォームの中央に移動します。
7. [valueButton1] をダブルクリックします。

   コード エディタが開き、valueButton1_Click イベントが表示されます。
8. 次のコード行を追加します。

```
// Visual J#
label1.set_Text("" + valueButton1.get_ButtonValue());
```

9. ソリューション エクスプローラで Test を右クリックし、ショートカット メニューの [スタートアップ プロジェクトに設定] をクリックします。
10. [デバッグ] メニューの [デバッグ開始] をクリックします。

    Form1 が表示されます。
11. [valueButton1] をクリックします。

    Label1 に数字 "5" が表示されます。これにより、`valueButton1_Click` メソッドによって継承コントロールの `ButtonValue` メンバが Label1 に渡されたことが証明されます。これで、作成した ValueButton コントロールが標準 Windows フォーム ボタンのすべての機能を継承し、それに加えてカスタム メンバを公開することがわかります。

参照

処理手順

チュートリアル : Visual J# でのユーザー コントロールの作成

その他の技術情報

コンポーネントによるプログラミング

コンポーネントおよびコントロール作成のチュートリアル

Visual J# リファレンス

このセクションでは、Visual J# プログラミングに関するさまざまなリファレンスへのリンクを示します。

このセクションの内容

言語サポート

Visual J++® 6.0 と .NET Framework の、Visual J# でサポートされている機能とサポートされていない機能に関するリファレンスです。

クラス ライブラリのサポート

パッケージのサポートの有無、およびクラス ライブラリのサポートに関する実装固有の詳細情報の一覧です。

Visual J# で記述されたアプリケーションでのセキュリティのセマンティクス

Visual J# アプリケーションのセキュリティに関する考慮事項について説明します。

Visual J# のアップグレード リファレンス

Visual J++ 6.0 のプロジェクトを Visual J# にアップグレードする場合の問題に関するリファレンスです。

J# のキーワード

Visual J# プログラムで使用できる言語キーワードに関するトピックへのリンクを示します。

J# の演算子

Visual J# プログラムで使用できる演算子に関するトピックへのリンクを示します。

Visual J# コンパイラ

Visual J# コンパイラ オプションについてのリファレンスです。コマンド ラインでのビルドに関する情報も記載されています。

Visual J# Class Library

J# クラス ライブラリについて紹介します。J# クラス ライブラリとは、JDK 1.1.4 仕様のサブセットの Microsoft による実装です。

関連するセクション

Visual J#

Visual J# ドキュメントのさまざまなトピックへのリンクを示します。

言語サポート

Visual J# コンパイラでは、デリゲート、J/Direct®、Java 言語/COM 相互運用機能用の **@com** 属性サポート、および条件付きコンパイルがサポートされており、**@security**、**@hidden** など他の属性の大半がサポートされています。また、Visual J# は、.NET Framework クラス ライブラリの使用をサポートするほか、Visual C#、Visual Basic などの .NET Framework 準拠の他の言語で記述されたユーザー定義クラス ライブラリもサポートしています。

このセクションの内容

Visual J++ 6.0 のサポート

Visual J# で完全にサポートされている、Visual J++ 6.0 の Microsoft 拡張機能の一覧です。

.NET Framework クラスの使用のサポート

Visual J# で、共通言語仕様 (CLS: Common Language Specification) に準拠する .NET Framework クラスをどのように使用できるかを示す簡単なサンプルが用意されています。

.NET Framework を対象とするための構文

.NET Framework 共通言語ランタイムの機能を使うための構文を示すトピックへのリンクの一覧です。

サポートされていない .NET Framework 機能

Visual J# でサポートされていない、.NET Framework 共通言語ランタイムの機能の一覧です。

チュートリアル : Java 言語でのソース コードの動的な生成とコンパイル

Visual J# に用意されている、Java 言語用の CodeDOM 実装について説明します。

Visual J# のドキュメント コメント

ドキュメント コメントをサポートする統合開発環境の機能について説明します。

Visual J# の例外の階層構造

Java 言語の例外と .NET Framework の例外との関係について説明します。

関連するセクション

Visual J# リファレンス

Visual J# コンパイラ、バイナリ変換ツール (JbImp.exe)、言語とライブラリのサポート、Visual J++ 6.0 プロジェクトのアップグレード、.NET Framework クラス ライブラリ用の Visual J# 構文、および開発環境の機能に関するリファレンス トピックです。

クラス ライブラリのサポート

パッケージのサポートの有無、およびクラス ライブラリのサポートに関する実装固有の詳細情報の一覧です。

Visual J++ 6.0 のサポート

このトピックで示す Visual J++ 6.0 の Microsoft 拡張機能は、Visual J# で完全にサポートされています。

拡張機能のリファレンスについては、MSDN オンライン ライブラリ (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vjcore98/html/vjovrdocumentationmap.asphttp://msdn.microsoft.com/library/default.asp?url=/library/en-us/vjcore98/html/vjovrdocumentationmap.asp) で、Visual J++ 6.0 ドキュメントまたは Microsoft SDK for Java を参照してください。

- J/Direct (ネイティブ DLL の呼び出し)
    - **@dll.import**
    - **@dll.struct**
    - **@dll.structmap**
- 次のような Java 言語/COM
    - **@com.class**
    - **@com.interface**
    - **@com.method**
    - **@com.parameter**
    - **@com.register**
- 条件付きコンパイル
    - **#if、#elif、#endif、#define**
    - **#warning、#error**
    - **@conditional**

      メソッドからの呼び出しを、生成されたコードから条件付きで削除できるメソッドを示すために使用します。

メモ：
@conditional 機能は、属性指定先の関数と呼び出し元が同じコンパイル ユニット内に存在する場合にのみ有効です。Visual J# で、条件付きコンパイルを行う場合は、.NET Framework の ConditionalAttribute クラスを使用することをお勧めします。

- デリゲートとマルチキャスト デリゲートのサポート
    - **com.ms.lang.Delegate** クラスおよび **com.ms.lang.MulticastDelegate** クラス
    - **delegate** キーワードおよび **multicast** キーワード
- その他の @ ディレクティブ
    - **@security**
    - **@hidden**
    - **@deprecated**

Visual J# コンパイラでは、Visual J++ 6.0 コンパイラの Java 言語のバイトコードに固有の以下の機能がサポートされていません。

- Java 言語のバイトコード (.class ファイル) からクラス情報をインポートする機能。
- Java 言語ソースから .class ファイルを生成する機能。
- コンパイル時に参照先クラスを解決するための **CLASSPATH** のサポート。代わりに /reference (インポート メタデータ) コンパイラ オプションを使います。

参照
その他の技術情報
言語サポート

.NET Framework クラスの使用のサポート

Visual J# コンパイラでは、共通言語仕様 (CLS: Common Language Specification) に準拠しているすべての .NET Framework ライブラリ API の使用がサポートされています。.NET Framework クラスを使うサンプル プログラム "Hello, World!" を次に示します。

使用例

```
// vjc_consume_netfx.jsl
// Contains the Console class.
import System.*;
public class Hello
{
    public static void main(String[] args)
    {
        Console.WriteLine("Hello, World!");
    }
}
```

出力

```
Hello, World!
```

参照

その他の技術情報

言語サポート

.NET Framework を対象とするための構文

このセクションでは、.NET Framework を対象とするために使用される Visual J# コンパイラの機能について説明します。特に指定しない限り、これらの機能は使用できるようになっています。ただし、/x (言語拡張機能の無効化) コンパイラ オプションを使って無効にすることができます。

このセクションでは、.NET Framework、共通型システム、値型、参照型、およびボックス化とボックス化解除を使った経験があることを前提としています。

このセクションでは、次の項目について説明します。

- オブジェクト階層のセマンティクス
- .NET Framework をサポートする言語拡張機能
    - 四角形配列
    - 新しいプリプロセッサ ディレクティブ
    - 属性の追加
    - カスタム属性の作成
    - 識別子としてのキーワードの使用
    - 参照渡しの引数を受け取るメソッドの呼び出し
    - 参照渡しの引数を受け取るメソッドの定義
- プロパティの定義と使用
- bean スタイルのプロパティ
- イベントの定義と使用
- .NET Framework デリゲートの定義と使用
- 値型の使用
- プリミティブ型に対応する値型の使用
- ユーザー定義の値型
- 列挙型の使用
- ユーザー定義の列挙型
- プロパティとイベントの公開
- ネイティブ メソッドの呼び出し
- 型変換演算
- .NET シリアル化のサポート
- 共通言語仕様への準拠の検証
- J# のジェネリック

参照

その他の技術情報

言語サポート

オブジェクト階層のセマンティクス

Visual J# では、**java.lang.Object** はオブジェクト階層の言語固有のルートとして扱われます。共通言語ランタイムによってサポートされている言語からこのオブジェクトにアクセスする場合は、Object を使用します。したがって、これらの言語では、Visual J# オブジェクトが、**java.lang.Object** から派生した特殊オブジェクトとはみなされません。

たとえば、Visual J# で記述されたクラスの場合、別の言語からそのオブジェクト メソッドに **toString** でアクセスできません。

```
class A extends java.lang.Object {
    public String toString() { return "Instance of A";}
}
```

オブジェクトの文字列形式を Visual C# で取得するには、代わりに、次のコードを使います。

```
string str = new A().ToString();
```

メモ：

これは、`A` が `toString` メソッドの実装をオーバーライドしていたとしても同じです。

Java 言語の開発者から見ると、Visual J# では、**Object** が **java.lang.Object** と同様に扱われ、機能が対応付けられているため、2 つのオブジェクトの違いを意識することはありません。これにより、代入やパラメータの受け渡しなどのために、**java.lang.Object** と **Object** との間で受け渡しを行うことができます。これは、**java.lang.String** および **Object** の場合も同じです。

Visual J# のバイナリを利用する方法は、Visual Basic や Visual C# で生成されたバイナリを利用する方法と同じです。

参照

その他の技術情報

言語サポート

.NET Framework をサポートする言語拡張機能

.NET Framework の使用をサポートする新機能を次に示します。

- 新しいプリプロセッサ ディレクティブ
- 属性の追加
- 識別子としてのキーワードの使用
- 参照渡しの引数を受け取るメソッドの呼び出し

参照

関連項目

.NET Framework を対象とするための構文

四角形配列

Visual J# は、ジャグ配列に加え、四角形配列もサポートしています。要素の型と配列の形状、たとえば次元の数などは、配列の型の一部です。ただし、各次元の長さで表される配列のサイズは、配列の型の一部ではありません。

この区別は、Visual J# 言語構文では明確になっています。つまり、各次元の長さは、配列の型ではなく、配列作成式の中で指定します。たとえば、`int[,,] arr = new int[1, 1, 2];` という宣言では、配列の型は `int[,,]` で、配列作成式は `new int[1, 1, 2]` です。

四角形配列とジャグ配列を併用することもできます。たとえば、`int [,][] mixedArr = new int[,][] {{{1,2}}, {{2,3,4}}};` は、混合された、ジャグ配列の 2 次元四角形配列です。この配列は、次元が `[2,1]` の四角形配列です。四角形配列の各要素は、1 次元の整数配列です。`mixedArr[0, 0]` での配列の長さは 2 で、`mixedArr[1, 0]` での配列の長さは 3 です。

使用例

Visual J# で四角形配列を作成する方法を示すサンプルを次に示します。

```
// vjc_rect_array.jsl
public class MyClass
{
    public static void main(String [] args)
    {
        // 2-D rectangular array; explicit creation.
        int[,] arr = new int[2, 2];
        arr[0, 0] = 1;
        arr[0, 1] = 2;
        arr[1, 0] = 3;
        arr[1, 1] = 4;

        // 2-D rectangular array;
        // Explicit creation with initializer list.
        int[,] arr2 = new int[,] {{1, 2}, {3, 4}};

        // 2-D rectangular array;
        // Shorthand: creation with initializer list.
        int[,] arr3 = {{1, 2}, {3, 4}};

        // 3-D rectangular array; explicit creation.
        int[,,] arr4 = new int[1, 1, 2];
        arr4[0, 0, 0] = 1;
        arr4[0, 0, 1] = 2;

        // 3-D rectangular array;
        // Explicit creation with initializer list.
        int[,,] arr5 = new int [,,]
            {{{1,2,3},{1,2,3}},{{1,2,3},{1,2,3}}};

        // Mix rectangle & jagged arrays; explicit creation using
        // new. Creates the same array as the above example.
        int [,][] mixedArr = new int[,][] {{{1,2}}, {{2,3,4}}};

        // Mixed array dynamic creation
        int [,][] mixedArr2 = new int[2,1][];
        mixedArr2[0,0] = new int[]{1,2};
        mixedArr2[1,0] = new int[]{2,3,4};
    }
}
```

参照

関連項目

.NET Framework を対象とするための構文

.NET Framework をサポートする言語拡張機能

Visual J# プリプロセッサ ディレクティブ

以下のプリプロセッサ ディレクティブを使用できます。

#endregion (Visual J#)

#region ブロックの終了を示します。

#line (Visual J#)

エラーや警告で出力するコンパイラの行番号を変更できます。

#pragma checksum (Visual J#)

アセンブリや .ASPX ファイルが変更されていないことを保証するためにチェックサム計算を適用します。

#region (Visual J#)

コードのブロックを指定できます。このブロックは、Visual Studio コード エディタのアウトライン機能を使って、展開や折りたたみができます。

参照

関連項目

.NET Framework を対象とするための構文

.NET Framework をサポートする言語拡張機能

#endregion (Visual J#)

#endregion は、**#region** ブロックの終了を示します。

```
#endregion
```

解説

#endregion の使用例については、「#region (Visual J#)」を参照してください。

参照

関連項目

Visual J# プリプロセッサ ディレクティブ

#line (Visual J#)

#line を使うと、エラーや警告でコンパイラが出力する行番号を変更できます。オプションでファイル名も変更できます。

```
#line [ number ["file_name"] | hidden | default ]
```

パラメータ

Number

ソース コード ファイルの行番号として指定する番号。このディレクティブの次の行に適用されます。

file_name (省略可能)

コンパイラ出力に表示するファイル名。既定では、ソース コード ファイルの実際の名前が使われます。ファイル名は、二重引用符 ("") で囲みます。ファイル パスを指定するときは、リテラル文字列で行う場合と同じように、実際の "\" 文字を区切るために区切り文字 "\" を使う必要があります。

hidden

別の **#line** ディレクティブが検出されるまで、後続の行をデバッガから隠します。

default

ファイルの行番号指定をリセットします。

解説

#line default ディレクティブは、行番号を元のソース行の番号に戻します。

#line hidden ディレクティブは、後続の行をデバッガで非表示にします。コードをたどるときに、**#line hidden** と、次に登場する **#line** ディレクティブの間の行は飛ばされます。ただし、間に別の **#line hidden** ディレクティブは指定されていないものとします。このオプションを ASP.NET で使用すると、ユーザー定義のコードとコンピュータが生成したコードを区別できます。この機能は、元々 ASP.NET で使用する機能ですが、他のソース ジェネレータでも使用できます。

#line hidden ディレクティブは、エラー報告のファイル名や行番号には影響しません。非表示のブロックにエラーがあった場合に、コンパイラは、エラーになったファイルと行番号を報告します。

ソース コード ファイルには、任意の数の **#line** ディレクティブを指定できます。

#line ディレクティブは、ビルド プロセスの、自動化された中間ステップで使用すると便利な場合があります。たとえば、中間ステップで元のソース コード ファイルから行を削除した場合でも、コンパイラがファイル内での削除前の行番号のままで出力を生成できるように、行を削除してから **#line** を指定して、削除前の行番号指定をシミュレートします。

ソース コード ファイルには、任意の数の **#line** ディレクティブを指定できます。

使用例

```
// preprocessor_line.jsl
// compile with: /W:3
public class MyClass2
{
    public static void main(String [] args)
    {
        #line 200
        int i;   // VJS1493 on line 200
        #line 8 "c:\\hashline\\another_file.jsl"
        char c;   // VJS1493 on line 8 of c:\hashline\another_file.jsl
        #line default
        int r; // VJS1493 on the original source line and file.
    }
}
```

参照

関連項目

Visual J# プリプロセッサ ディレクティブ

#pragma checksum (Visual J#)

生成されたソース ファイルと ASPX ファイルの **checksum** を計算します。

```
#pragma checksum "filename" "{guid}" "checksum_bytes"
```

パラメータ

"filename"

変更や更新を監視する必要のあるファイルの名前。

"{guid}"

ファイルのグローバルー意識別子 (GUID: Globally Unique Identifier)。

"*checksum_bytes*"

チェックサム対象のバイトを表す 16 進数の文字列。偶数桁の 16 進数である必要があります。奇数桁を指定すると、コンパイル時に警告が出力され、ディレクティブが無視されます。

例外

#pragma checksum ディレクティブに、値が指定されていないなどの構文エラーがあると、コンパイラから警告が出力され、ディレクティブが無視されます。

解説

Visual Studio デバッガでは、**checksum** を使用して、常に正しいソースが検索されます。コンパイラは、ソース ファイルの **checksum** を計算し、結果を PDB に出力します。次にデバッガが PDB を参照し、デバッガでソース ファイルから計算した **checksum** と比較します。

この方法は ASP.NET プロジェクトでは使用できません。**checksum** が ASPX ファイルではなく、生成されたソース ファイルから計算されるためです。この問題を解決するために、**#pragma checksum** では ASP .NET ページを対象に **checksum** を計算する機能がサポートされています。

Visual J# で ASP.NET プロジェクトを作成すると、生成されるソース ファイルには、そのファイルの作成元の .ASPX ファイルに対する **checksum** が入っています。コンパイラは、この情報を PDB ファイルに書き込みます。

ファイル内に **#pragma checksum** ディレクティブが指定されていない場合、コンパイラは **checksum** を計算して、その値を PDB ファイルに書き込みます。

説明

ASP.NET プロジェクトを作成した場合、一般的なソース ファイルの先頭には、以下のディレクティブが記述してあります。

```
#pragma checksum "MyFileName" "{XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}" "XXXX...."
```

参照

その他の技術情報

ASP.NET のプログラミング

#region (Visual J#)

コードのブロックを指定できます。このブロックは、Visual Studio コード エディタのアウトライン機能を使って、展開や折りたたみができます。

```
#region name
```

パラメータ
name

Visual Studio コード エディタに表示される領域に指定する名前。

解説

#region ブロックは、#endregion (Visual J#) ディレクティブで終了させる必要があります。

#region ブロックは、**#if** ブロックとオーバーラップすることはできません。ただし、**#region** ブロックを **#if** ブロック内に入れ子にしたり、**#if** ブロックを **#region** ブロック内に入れ子にしたりすることはできます。

Visual J# コンパイラでは、**#region** や **#endregion** で始まる行をコメントとして扱うため、これらのディレクティブはコマンド ライン ビルドで無視されます。

使用例

```
// preprocessor_region.jsl
#region MyClass definition
public class MyClass
{
   public static void main(String [] args)
   {
   }
}
#endregion
```

参照
関連項目
Visual J# プリプロセッサ ディレクティブ

属性の追加

Visual J# では、クラス、フィールド、メソッド、パラメータ、およびその他のプログラミング要素に属性を追加できます。カスタム属性および擬似属性 (リフレクションによって照会できない属性) の両方をメタデータに追加できます。Visual J# では、メタデータに属性を追加できるようにするために、Visual J++ の拡張構文に従います。

属性を追加するには、**@attribute** ディレクティブに続けて、追加する属性の名前を指定します。属性パラメータが任意の型オブジェクトを受け入れる場合には、基になるクラス オブジェクトを型オブジェクトにキャストするのではなく、そのまま引数として渡す必要があります。

@attribute.return ディレクティブを使うと、戻り値に属性が追加されます。**@attribute.method** ディレクティブを使うと、基になるメソッドの属性を指定できます。

これらのディレクティブは、指定されたもの以外のプログラミング オブジェクトに追加するとエラーになります。他の属性の場合は、ディレクティブの後に指定されたメンバを基にして、コンパイラによって追加先が決定されます。

使用例

次の例は、**@attribute** を使用する方法を示しています。

```
// vjc_attributes1.jsl
// compile with: /target:library
import System.*;
import System.Runtime.InteropServices.*;
public class MyClass
{
    /** @attribute DllImport("user32", CharSet=CharSet.Ansi) */
    public static native int MessageBox(
        int hwnd,
        /** @attribute MarshalAs(UnmanagedType.AnsiBStr)*/
        System.String title,
        System.String caption,
        int type);
}
```

@attribute.return を使用して戻り値の型のマーシャリング属性を指定する方法を次の例に示します。

```
// vjc_attributes2.jsl
// compile with: /target:library
import System.Runtime.InteropServices.*;
public class MyClass
{
    /** @attribute.return MarshalAs(UnmanagedType.LPArray, SizeConst=10) */
    int[] getBytes()
    {
        int arr[] = {1,2};
        return arr;
    }
}
```

@attribute.method を使用して戻り値の型のマーシャリング属性を指定する方法を次の例に示します。

```
// vjc_attributes3.jsl
// compile with: /target:library
import System.*;
public class MyClass
{
    /** @property */
    /** @attribute.method Obsolete("get_Bytes is deprecated", false) */
    // The directive ensures that the attribute is attached to the method.
    int[] get_Bytes()
    {
        int arr[] = {1,2};
        return arr;
    }
```

```
}
```

次の例は、属性宣言内で **.class** 構文を使用し、Type を取得する方法を示しています。

```
/** @attribute System.Xml.Serialization.XmlArrayItem(Simple.class)) */
```

Visual J# プログラムでは、属性を定義できません。.NET Framework やその他の参照先アセンブリの属性を使用できるだけです。

アセンブリ レベルの属性は、**@attribute** ディレクティブではなく **@assembly** ディレクティブを使って追加されます。**@assembly** ディレクティブは、ファイルの先頭に配置する必要があります。ただし、ファイルの先頭に package ステートメントや import ステートメントがある場合は、それらの後に指定します。

```
import System.Reflection.*;
/** @assembly AssemblyTitle("My Assembly") */
/** @assembly AssemblyCompany("My Company") */
```

属性の文法

属性の文法を次に示します。

用語	定義
Attributes	*attribute-sections*
attribute-sections	*attribute-section* *attribute-sections attribute-section*
attribute-section	*/** attribute-list **/*
attribute-list	newlineopt *attribute* newlineopt *attribute-list* newline *attribute*
Attribute	*attribute-target-specifier attribute-name attribute-arguments*
attribute-target-specifier	**@attribute** **@assembly** **@attribute.return** **@attribute.method**
attribute-name	*type-name*
attribute-arguments	(*positional-argument-list*opt) (*positional-argument-list* , *named-argument-list*) (*named-argument-list*)
positional-argument-list	*positional-argument* *positional-argument-list* , *positional-argument*
positional-argument	*attribute-argument-expression*
named-argument-list	*named-argument* *named-argument-list* , *named-argument*

named-argument	*attribute-sections*
attribute-argument-expression	*attribute-section* *attribute-sections attribute-section*

属性追加の規則

- 属性は、クラス、フィールド、メソッド、またはメソッド パラメータに追加できます。
- 属性は、行の先頭から開始する必要があります。
- コメントの開始位置に続くリスト内のアスタリスク (*) は、空白として扱われます。同様に、キーワード間や引数間の改行文字 (\n) も空白と見なされます。
- コンマは引数の間を区切ります。
- *@attribute-target-specifier* に続く型名は、現在のコンテキスト内の属性型に解決される必要があります。クラス名は、完全限定名で指定するか、または単に簡易名で指定することもできます。簡易名の場合は、型を解決するためにインポート情報が使用されます。同様に、名前付き引数の *identifier* は、属性型のプロパティまたはフィールドに対応している必要があります。
- 規約により、属性クラスの名前には Attribute というサフィックスを付けます。*type-name* 形式の *attribute-name* では、このサフィックスを含めることも省略することもできます。

```
/** @attribute STAThreadAttribute() */ // refers to STAThreadAttribute
public static void main(String [] args) {}
// The Attribute suffix is optional
/** @attribute STAThread() */ // Also refers to STAThreadAttribute
public static void main(String [] args) {}
```

- 属性に対するパラメータが Type オブジェクトの場合には、基になるクラス オブジェクトを引数として渡す必要があります。

参照

関連項目

.NET Framework を対象とするための構文

.NET Framework をサポートする言語拡張機能

カスタム属性の作成

Visual J# では、独自の .NET Framework 属性を作成できます。これを行うには、.NET Framework の Attribute クラスを拡張します。詳細については、「属性の追加」を参照してください。

カスタム属性の宣言は、クラスの宣言に類似していますが、以下の規則が適用されます。

- カスタム属性は、**Attribute** を明示的に拡張する必要があります。
- カスタム属性では、定義済みの AttributeUsageAttribute 属性を使用して属性の特性を定義する必要があります。
- 属性クラス内の名前付きフィールドおよびプロパティは、パブリックである必要があります。
- カスタム属性を内部クラスにすることはできません。
- カスタム属性を匿名クラスにすることはできません。
- 規則により、カスタム属性クラスには "Attribute" で終わる名前を付ける必要があります。属性を割り当てるときは、この "Attribute" サフィックスを省略できます。

AttributeUsageAttribute の使用

AttributeUsageAttribute は次のように使用します。

```
/** @attribute System.AttributeUsage(
AttributeTargets.All,
Inherited = false,
AllowMultiple = true
) */
```

最初の引数は、この属性をどのコンテキストで使用するのが有効であるのかを示す AttributeTargets 列挙値です。次の値を指定できます。複数のターゲットを指定するために、ビットごとの OR 演算子を使用できます。

メンバ名	属性の適用対象
All	すべてのプログラム要素
Assembly	アセンブリ
Class	クラス
Constructor	コンストラクタ
Delegate	デリゲート
Enum	列挙体
Event	イベント
Field	フィールド
Interface	インターフェイス
Method	メソッド
Module	モジュール
Parameter	パラメータ

Property	プロパティ
Return	戻り値
Struct	ユーザー定義の値型

2 番目の引数 Inherited は、この属性をクラスに割り当てる場合に、適用されるクラスを拡張するクラスによってこの属性が継承されるかどうかを示します。

3 番目の引数 AllowMultiple は、同じエンティティにこの属性のインスタンスを複数割り当てることができるかどうかを示します。詳細については、「属性の追加」を参照してください。

属性クラスにはコンストラクタがあり、その引数は、この属性をターゲットに割り当てる際に提供されます。これには、いくつかの制限があります。2D 以上の配列は、属性コンストラクタのパラメータで使用できません。パラメータは CLS 準拠の型である必要があります。属性を使用する場合、引数は定数である必要があります。

名前付きフィールドおよびプロパティ

位置指定パラメータは、属性コンストラクタの引数です。これらは必須であり、属性を使用するたびに指定する必要があります。名前付きパラメータは、属性コンストラクタで定義されているのではなく、パブリック フィールドまたはプロパティです。名前付きパラメータでは、属性がインスタンス化される場合に、クライアントは属性のフィールドおよびプロパティを設定できます。パブリックな各 CTOR では、他の各種クラスと同様、一連の位置指定パラメータを定義できます。ただし、属性の場合、位置指定パラメータを一度指定した後は、次の構文を使用してフィールドまたはプロパティを参照できます。

```
FieldOrPropertyName=value
```

解説

この方法で設定できるようにするには、フィールドおよびプロパティはパブリックである必要があります。

属性の作成を無効にする

/x:net オプションを使用すると、属性の作成が無効になります。

使用例

```
// CustomAttribute.jsl
import System.*;
import System.Diagnostics.*;

/** @attribute AttributeUsage(AttributeTargets.All,
                              Inherited = false,
                              AllowMultiple = false) */
public class OwnerAttribute extends System.Attribute
{
    private String m_owner;
    private String m_phone;

    public OwnerAttribute()
    {
      m_owner = "";
      m_phone = "";
    }

    public OwnerAttribute(String owner, String phone)
    {
       m_owner = owner;
       m_phone = phone;
    }

    /** @property */
    public String get_Owner()
    {
       return m_owner;
    }

    /** @property */
```

```
    public String get_Phone()
    {
       return m_phone;
    }
}

/** @attribute Owner("JohnS", "555-3890") */
public class C1
{ }

/** @attribute Owner("EmilyK", "555-2121") */
public class C2 extends C1
{ }

class CMain
{

    public static void main()
    {
      C1 c1 = new C1();
      C2 c2 = new C2();
      OwnerAttribute oa = new OwnerAttribute();
      // GetCustomAttribute may be used to get attributes attached to a
      // type. The function takes two arguments of System.Type
      // representing the class type and the attribute type.
      oa = (OwnerAttribute) Attribute.GetCustomAttribute(c1.GetType(),
                                                         oa.GetType());
      Console.WriteLine("C1 Owner: {0} Phone: {1}", oa.get_Owner(),
                                                    oa.get_Phone());
      oa = (OwnerAttribute) Attribute.GetCustomAttribute(c2.GetType(),
                                                         oa.GetType());
      Console.WriteLine("C2 Owner: {0} Phone: {1}", oa.get_Owner(),
                                                    oa.get_Phone());
    }
}
```

出力

```
C1 Owner: JohnS Phone: 555-3890
C2 Owner: EmilyK Phone: 555-2121
```

参照

関連項目

属性の追加

.NET Framework を対象とするための構文

.NET Framework をサポートする言語拡張機能

識別子としてのキーワードの使用

Visual J# では、**@keyword** を識別子として扱うことができ、Visual J# によって予約されたキーワードである参照先アセンブリで識別子を使用できます。

使用例

```
// vjc_identifier_1.cs
// compile with: /target:library
// C# program
public class MyClass1
{
   public static int synchronized = 0;   // valid in C#
}
// vjc_identifier_2.jsl
// compile with: /reference:vjc_identifier_1.dll
public class MyClass2
{
    public static void main(String [] args)
    {
        MyClass1 x = new MyClass1();
        // System.Console.WriteLine(x.synchronized);   // error
        System.Console.WriteLine(x.@synchronized);   // OK
    }
}
```

出力

```
0
```

参照

関連項目

.NET Framework を対象とするための構文

.NET Framework をサポートする言語拡張機能

参照渡しの引数を受け取るメソッドの呼び出し

.NET Framework では、パラメータの参照渡しがサポートされています。参照渡しでは、関数内部でパラメータを変更でき、呼び出し元で変更内容を参照できます。Visual J# では、C++ スタイルに従います。C++ スタイルでは、メソッドを直接呼び出してパラメータを渡すことができ、Visual J# コンパイラが暗黙の変換を行います。

使用例

```
// vjc_args_by_ref_1.cs
// compile with: /target:library
// A C# program used as a DLL
public class MyClass1
{
    public void Test(ref string str)
    {
        str += " World!";
    }
}
```

Java 言語では、このメソッドを次の方法で呼び出すことができます。

```
// vjc_args_by_ref_2.jsl
// compile with: /reference:vjc_args_by_ref_1.dll
public class MyClass2
{
    public static void main(String [] args)
    {
        MyClass1 x = new MyClass1();
        System.String mystr = "Hello";
        System.Console.WriteLine(mystr);
        x.Test(mystr);
        System.Console.WriteLine(mystr);
    }
}
```

出力

```
Hello
Hello World!
```

参照

関連項目

.NET Framework を対象とするための構文
.NET Framework をサポートする言語拡張機能

参照渡しの引数を受け取るメソッドの定義

通常、関数の引数は、値渡しされます。値型の場合、関数は渡されたオブジェクトのコピーを作成して、このコピーを操作します。したがって、元のオブジェクトの値が関数によって変更されることはありません。参照型の場合、関数は参照先のオブジェクトを直接操作します。したがって、参照先オブジェクトを変更できます。一方、値で渡された場合には、参照先の変数そのものを変更することはできません。

参照渡しで値型を渡した場合、関数は渡されたオブジェクトそのものを操作することを意味します。したがって、関数内で行われた変更は元のオブジェクトにすべて反映されます。参照渡しで参照型を渡した場合、関数は参照先を他のオブジェクトや **null** に変更できることを意味します。

関数宣言では、パラメータに適用される **/** @ref */** タグを使用することによって、参照渡しのパラメータであることを宣言できます。このタグは、次の例で示すように、このパラメータに適用する最初の修飾子にしてください。

```
void f(/** @ref */ int i );
```

/ @ref */** タグは、プリミティブ型、配列、ユーザー定義の値型、列挙型および他の参照型に適用できます。メソッドを呼び出すときに、引数にタグを付けるとエラーになります。

パラメータを参照渡しするかどうかだけが異なるメソッドを、オーバーロードする宣言はエラーになります。呼び出し形式で 2 つのメソッドを区別することはできません。

/ @ref */** タグは、/x:net オプションと一緒に使用できません。

使用例

```
// ByRef.jsl
import System.*;

public class C
{
   public static final int MAX = 10;

   public static void Increment(/** @ref */ int i)
   {
      i++;
   }

   public static void main()
   {
       for (int i = 0; i <= MAX; C.Increment(i))
       {
           Console.WriteLine("Value: " + i);
       }
   }
}
```

出力

```
Value: 0
Value: 1
Value: 2
Value: 3
Value: 4
Value: 5
Value: 6
Value: 7
Value: 8
Value: 9
Value: 10
```

参照

処理手順

参照渡しの引数を受け取るメソッドの呼び出し

関連項目

.NET Framework を対象とするための構文

.NET Framework をサポートする言語拡張機能

プロパティの定義と使用

Visual J# では、.NET Framework クラスで定義されているプロパティにアクセスできます。Visual J# は、.NET Framework クラスのプロパティを **get_** アクセサ メソッドと **set_** アクセサ メソッドに変換します。Visual J# を使う場合は、アクセサ メソッドを直接呼び出します。Visual J# では、プロパティ アクセサに特別な重要性が追加されることはありません。プロパティ アクセサは、クラスのメソッドとして扱われます。

使用例

```
// vjc_properties.jsl
// compile with: /reference:System.Windows.Forms.dll
import System.Windows.Forms.*;
public class MyClass
{
    public static void main(String [] args)
    {
        Button b = new Button();
        String str = b.get_Text();
        System.Console.WriteLine("Button Text = " + str);
        b.set_Text("Sample Button");
        str = b.get_Text();
        System.Console.WriteLine("Button Text = " + str);
    }
}
```

```
public class MyProp
{
    private int i = -1;

    /** @property */
    public int get_SomeNum()
    {
        return i;
    }

    /** @property */
    public void set_SomeNum(int input)
    {
        i = input;
    }

    public static void main()
    {
        MyProp aprop = new MyProp();
        System.Console.WriteLine(aprop.get_SomeNum());
        aprop.set_SomeNum(8);
        System.Console.WriteLine(aprop.get_SomeNum());
    }
}
```

同じ名前の 2 つのプロパティを作成するサンプルを次に示します。

```
public class PropTest
{
    private int i = -1;
    private int j = -2;

    /** @property */
    public int get_SomeNum()
    {
        return i;
    }

    /** @property */
    public void set_SomeNum(int input)
```

```
    {
        i = input;
    }

    /** @property */
    public int get_SomeNum(int i)
    {
        return j;
    }

    /** @property */
    public void set_SomeNum(int i, int input)
    {
        j = input;
    }

    public static void main(String [] args)
    {
        int i = 0;
        PropTest aprop = new PropTest();
        System.Console.WriteLine(aprop.get_SomeNum());
        System.Console.WriteLine(aprop.get_SomeNum(i));
        aprop.set_SomeNum(8);
        aprop.set_SomeNum(i,9);
        System.Console.WriteLine(aprop.get_SomeNum());
        System.Console.WriteLine(aprop.get_SomeNum(i));
    }
}
```

次の例は、1 つのプロパティを持つ Visual J# コンポーネントを示しています。続けて、このプロパティを Visual J# クライアントで使う方法を示し、次に C# クライアントで使う方法も示します。

```
// vjc_prop_use.jsl
// compile with: /target:library
public class PropTest
{
    private int i = -1;

    /** @property */
    public int get_SomeNum()
    {
        return i;
    }

    /** @property */
    public void set_SomeNum(int input)
    {
        i = input;
    }
}
```

次の例では、プロパティを使う Visual J# クライアントを示します。このサンプルをコンパイルするには、vjs_prop_use.dll を参照します。

```
// Consumer_1.jsl
// compile with: /reference: vjs_prop_use.dll
public class Test
{
    public static void main(String [] args)
    {
        int i = 0;
        PropTest aprop = new PropTest();
        System.Console.WriteLine(aprop.get_SomeNum());
        aprop.set_SomeNum(8);
        System.Console.WriteLine(aprop.get_SomeNum());
    }
}
```

次の例では、プロパティを使う Visual C# クライアントを示します。このサンプルをコンパイルするには、vjs_prop_use.dll を参照します。

```
// Consume2.cs
// compile with: /reference: vjs_prop_use.dll
public class Test
{
    public static void Main()
    {
        PropTest aprop = new PropTest();
        System.Console.WriteLine(aprop.SomeNum);
        aprop.SomeNum = 8;
        System.Console.WriteLine(aprop.SomeNum);
    }
}
```

インデックス付きプロパティを定義する Visual J# コンポーネントの例を次に示します。後に続く Visual C# サンプルは、このプロパティの使い方を示しています。

```
// vjc_indexed_properties.jsl
// compile with: /target:library
import System.*;
import System.Reflection.*;
class Employee
{
    public Employee(System.String s, int d)
    {
        _name = s;
        _dept = d;
    }

    /** @property */
    System.String get_name()
    {
        return _name;
    }

    /** @property */
    int get_dept()
    {
        return _dept;
    }

    private System.String _name;
    private int _dept;
}

/** @attribute DefaultMember("Report") */
// Sets the Report property as the indexer for the class.
class Manager
{
    /** @property */
    public Employee get_Report(System.String s)
    {
        for (pEmp = Reports ; (pEmp!=null) && (pEmp.emp.get_name() != s)
            ;
        pEmp = pEmp.next);
        if (pEmp!=null)
            return pEmp.emp;
        else
            return null;
    }

    /** @property */
    public void set_Report(System.String s, Employee e)
    {
        for (pEmp = Reports ; (pEmp!=null) && (pEmp.emp.get_name() != s)
```

```
            ;
        pEmp = pEmp.next);
        if (pEmp==null)
        {
            EmpList emp1 = new EmpList();
            emp1.emp = e;
            emp1.next = Reports;
            Reports = emp1;
        }
    }

    private static class EmpList     {
        public Employee emp;
        public EmpList next;
    }

    EmpList pEmp;
    static EmpList Reports = null;
}
```

次の Visual C# サンプルでは、前のサンプルで定義したプロパティを使います。このサンプルをコンパイルするには、reference vjc_indexed_properties.dll を参照します。

```
// Consume11.cs
// compile with: /reference vjc_indexed_properties.dll
using System;

public class Class1
{
    public static void Main()
    {
        Manager  Ed = new Manager();
        Employee Bob = new Employee("Bob Smith", 12);

        // track Ed's reports
        Ed[Bob.name] = Bob;    // indexed by string type
        Console.WriteLine(Ed[Bob.name].dept);
    }
}
```

特に指定しない限り、アクセサ メソッドに追加した属性は、対応するプロパティに適用されます。ただし、次のように **@attribute.method** ディレクティブを使うと、個々のメソッドに属性を追加できます。

```
// vjc_properties_and_attributes.jsl
// compile with: /target:library
import System.ComponentModel.*;

/*
 * Different attributes to getter and setter methods
 */
public class Scalar
{
    private int magnitude = 101;

    /**@attribute Description("Attribute on the property")*/
    /**@attribute.method Description("Attribute on get accessor")*/
    /**@property*/
    public int get_Magnitude()
    {
        return magnitude;
    }

    /**@attribute.method Description("Attrib on set accessor")*/
    /** @property */
    public void set_Magnitude(int value)
    {
```

```
            magnitude = value;
        }
    }
```

出力

```
Button Text =
Button Text = Sample Button
```

Visual J# では、プロパティの定義もサポートされています。.NET 構文と bean 構文の 2 とおりの構文がサポートされています。bean スタイルの構文については、「bean スタイルのプロパティ」を参照してください。プロパティのアクセサ メソッドを .NET 構文を使用して定義する場合、アクセサ メソッド名は **get_** または **set_** で始まる必要があります。プロパティは、下の例で示すように **@property** タグを使用して、プロパティとして識別できるようにする必要があります。

プロパティの各アクセサ メソッドには、同じアクセス修飾子を使う必要があります。たとえば、両方を **public** にするか、両方を **private** にします。プロパティの型は、**get_** アクセサ メソッドの戻り値の型、および **set_** アクセサ メソッドの最後の引数の型と同じである必要があります。

派生クラスでプロパティ アクセサ メソッドをオーバーライドできます。

Visual J# の場合、単純な .Net スタイルのプロパティは次のように定義されます。

出力

```
-1
8
```

プロパティは、適切なアクセサ (**get_** または **set_**) だけを定義することによって、読み取り専用または書き込み専用として定義できます。プロパティは、メソッド アクセサが言語規則と競合しない限りオーバーロードできます。

出力

```
-1
-2
8
9
```

出力

```
-1
8
```

出力

```
-1
8
```

出力

```
12
```

参照

関連項目

.NET Framework を対象とするための構文
プロパティとイベントの公開

bean スタイルのプロパティ

bean スタイルのプロパティでは、Java Beans で使用されている名前付けパターンを使用します。

解説

J# コードでプロパティを宣言する場合に、同義的に使用される、全く異なる 2 種類の名前付けパターンがあります。.NET 名前付けパターンでは、プレフィックスとして **get_** および **set_** を使用します。bean 名前付けパターンでは、プレフィックスとして **get**、**set**、および **is** を使用します。bean スタイルのプロパティを宣言する場合は、**/** @property */** タグの代わりに **/** @beanproperty */** タグを使用して、アクセサで bean スタイルの名前付けパターンが使用されていることを示します。コンパイラは、**get**、**set**、または **is** で始まるアクセサを検索します。コンパイラにより、**get** プレフィックス、**set** プレフィックス、または **is** プレフィックスに続く識別子の部分から、プロパティ名が推定されます。プロパティ名の最初の文字は、Java のメソッドを Camel 形式の組み合わせで記述するために、小文字に変換されます。ただし、プロパティ名がすべて大文字で記述されている場合を除きます。

例 1

`value` というプロパティを宣言する **/** @beanproperty */** タグの定義と、C# でそのプロパティを使用する場合の例を次に示します。

```
// beanprop1.jsl
// compile with: /target:module
public class SimpleProperty
{
  /** @beanproperty */
  public int getvalue() { return _value; }

  /** @beanproperty */
  public void setvalue (int v) { _value = v; }

  private int _value;
}
```

```
// bpconsume.cs
// compile with: /addmodule:beanprop1.netmodule /platform:x86
class CMain
{
  public static void Main()
  {
     SimpleProperty simple = new SimpleProperty();
     simple.value = 15;
     System.Console.WriteLine( simple.value );
  }
}
```

出力

```
15
```

例 2

/ @beanproperty */** タグを使用して `initialized` というブール型のプロパティを宣言する例を次に示します。

```
// beanprop2.jsl
// compile with: /target:module
/* Declare a read-only boolean bean property using
   the isXXXX pattern */

class BooleanProperty
{
   private boolean _init = false;

   /** @beanproperty */
   public boolean isinitialized()
   {
      return _init;
```

```
    }

    public void Initialize()
    {
       _init = true;
    }
}
```

```
// bpconsume2.cs
// compile with: /addmodule:beanprop2.netmodule /platform:x86
using System;

class CMain
{
   public static void Main()
   {
       BooleanProperty bp = new BooleanProperty();
       Console.WriteLine("Initialized: {0}", bp.initialized) ;
       bp.Initialize();
       Console.WriteLine("Initialized: {0}", bp.initialized) ;
   }
}
```

出力

```
Initialized: False
Initialized: True
```

参照

関連項目

プロパティの定義と使用

イベントの定義と使用

Visual J# では、.NET Framework クラスで発行されるイベントを登録できます。AddValueChanged イベント アクセサおよび RemoveValueChanged イベント アクセサを次コード例のように使用して、イベント ハンドラを登録できます。

```
import System.Windows.Forms.*;

public class MyClass
{
    public static void main(String [] args)
    {
        System.Windows.Forms.Button button =
            new System.Windows.Forms.Button();
        MyClass myClass = new MyClass();
        System.EventHandler eventhandler =
            new System.EventHandler(myClass.OnDoubleClick);
        button.add_DoubleClick(eventhandler);
        // OnDoubleClick is a method defined in the current class
        // that has the same signature as the EventHandler delegate.
        // The code above demonstrates how to use .NET Framework
        // delegates in Visual J#.

        // Similarly, you can unsubscribe to the event as follows:
        button.remove_DoubleClick(eventhandler);
    }

    private void OnDoubleClick(System.Object sender, System.EventArgs e)
    {
    }
}
```

Visual J# では、.NET Framework イベントの定義がサポートされています。

イベント宣言の形式を次に示します。

```
/** @event */
void add_EventName(event_type event)

/** @event */
void remove_EventName(event_type event)
```

上記の例の `event_type` は、イベントの型を定義するパラメータであり、Delegate から派生している必要があります。イベントの `add` メソッドおよび `remove` メソッドは、共に存在するか、共に存在しない必要があり、存在する場合には、同じアクセス修飾子が指定されている必要があります。

次のように、派生クラスでイベント アクセサ メソッドをオーバーライドできます。

```
import System.*;
import System.Collections.ArrayList;

/** @delegate */
public delegate void MyDelegate();   // delegate declaration

public interface I
{
    /** @event */
    public void add_MyEvent(MyDelegate listener);

    /** @event */
    public void remove_MyEvent(MyDelegate listener);

    void FireAway();
}
```

```
public class MyClass implements I
{
    private ArrayList list = new ArrayList(10);
    private int no = 0;

    /** @event */
    public void add_MyEvent(MyDelegate listener)
    {
        list.Add(listener);
    }

    /** @event */
    public void remove_MyEvent(MyDelegate listener)
    {
        list.Remove(listener);
    }

    public void FireAway()
    {
        System.Object [] toArray = list.ToArray();

        int len = toArray.length;
        for (int i = 0;i < len ;i++)
        {
            ((MyDelegate)(toArray[i])).Invoke();
        }
    }
}

public class MainClass
{
    static public void main (String [] args)
    {
        new MainClass();
    }

    public void f1()
    {
        Console.WriteLine("This is called when the event fires.");
    }

    public MainClass()
    {
        I i = new MyClass();

        i.add_MyEvent(new MyDelegate(this.f1));
        i.FireAway();
    }
}
```

前のコードの出力は、次のようになります。

```
This is called when the event fires.
```

前のプログラムが .DLL としてコンパイルされた場合、イベントには Visual C# プログラムから次のようにアクセスできます。

例 1

この例の `MainClass` では、`MyDelegate` からプライベートな `f()` メソッドを実行できます。

```
// vjc_events3.jsl
// compile with:  /t:library  /out:VJC_Events3.dll

import System.*;
import System.Collections.ArrayList;
```

```
/** @delegate */
public delegate void MyDelegate();   // delegate declaration

public interface I
{
    /** @event */
    public void add_MyEvent(MyDelegate listener);

    /** @event */
    public void remove_MyEvent(MyDelegate listener);

    void FireAway();
}

public class MyClass implements I
{
    private ArrayList list = new ArrayList(10);
    private int no = 0;

    /** @event */
    public void add_MyEvent(MyDelegate listener)
    {
        list.Add(listener);
    }

    /** @event */
    public void remove_MyEvent(MyDelegate listener)
    {
        list.Remove(listener);
    }

    public void FireAway()
    {
        System.Object[] toArray = list.ToArray();

        int len = toArray.length;
        for (int i = 0; i < len; i++)
        {
            ((MyDelegate)(toArray[i])).Invoke();
        }
    }
}
```

```
// vjc_events4.cs
// compile with: /reference:vjc_events3.dll
using System;

public class MainClass
{
    static private void f()
    {
        Console.WriteLine("This is called when the event fires.");
    }

    static public void Main ()
    {
        I i = new MyClass();

        i.MyEvent += new MyDelegate(f);
        i.FireAway();
    }
}
```

出力

```
This is called when the event fires.
```

参照

関連項目

.NET Framework を対象とするための構文

プロパティとイベントの公開

bean スタイルのプロパティ

bean スタイルのプロパティでは、Java Beans で使用されている名前付けパターンを使用します。

解説

J# コードでプロパティを宣言する場合に、同義的に使用される、全く異なる 2 種類の名前付けパターンがあります。.NET 名前付けパターンでは、プレフィックスとして **get_** および **set_** を使用します。bean 名前付けパターンでは、プレフィックスとして **get**、**set**、および **is** を使用します。bean スタイルのプロパティを宣言する場合は、**/** @property */** タグの代わりに **/** @beanproperty */** タグを使用して、アクセサで bean スタイルの名前付けパターンが使用されていることを示します。コンパイラは、**get**、**set**、または **is** で始まるアクセサを検索します。コンパイラにより、**get** プレフィックス、**set** プレフィックス、または **is** プレフィックスに続く識別子の部分から、プロパティ名が推定されます。プロパティ名の最初の文字は、Java のメソッドを Camel 形式の組み合わせで記述するために、小文字に変換されます。ただし、プロパティ名がすべて大文字で記述されている場合を除きます。

例 1

`value` というプロパティを宣言する **/** @beanproperty */** タグの定義と、C# でそのプロパティを使用する場合の例を次に示します。

```
// beanprop1.jsl
// compile with: /target:module
public class SimpleProperty
{
  /** @beanproperty */
  public int getvalue() { return _value; }

  /** @beanproperty */
  public void setvalue (int v) { _value = v; }

  private int _value;
}
```

```
// bpconsume.cs
// compile with: /addmodule:beanprop1.netmodule /platform:x86
class CMain
{
  public static void Main()
  {
     SimpleProperty simple = new SimpleProperty();
     simple.value = 15;
     System.Console.WriteLine( simple.value );
  }
}
```

出力

```
15
```

例 2

/ @beanproperty */** タグを使用して `initialized` というブール型のプロパティを宣言する例を次に示します。

```
// beanprop2.jsl
// compile with: /target:module
/* Declare a read-only boolean bean property using
   the isXXXX pattern */

class BooleanProperty
{
   private boolean _init = false;

   /** @beanproperty */
   public boolean isinitialized()
   {
      return _init;
```

```
    }

    public void Initialize()
    {
       _init = true;
    }
}
```

```
// bpconsume2.cs
// compile with: /addmodule:beanprop2.netmodule /platform:x86
using System;

class CMain
{
   public static void Main()
   {
       BooleanProperty bp = new BooleanProperty();
       Console.WriteLine("Initialized: {0}", bp.initialized) ;
       bp.Initialize();
       Console.WriteLine("Initialized: {0}", bp.initialized) ;
   }
}
```

出力

```
Initialized: False
Initialized: True
```

参照

関連項目

プロパティの定義と使用

.NET Framework デリゲートの定義と使用

Visual J# コンパイラでは、Visual J++ のデリゲート拡張機能と同様に、.NET Framework デリゲートを作成できます。重要な違いは、バインドが常にコンパイル時に行われることです。

使用例

```
import System.Windows.Forms.*;
import System.*;

public class MyObjClass
{
    public void OnDoubleClick(System.Object o, System.EventArgs e)
    {
        System.Console.WriteLine("test");
    }
}

public class MyClass
{
    public static void main()
    {
        MyObjClass obj = new MyObjClass();

        // Where OnDoubleClick is a method in obj.
        EventHandler handler = new EventHandler(obj.OnDoubleClick);
        handler.Invoke(obj, new EventArgs());
        // The following is also valid.
        EventHandler handler2 = new EventHandler(obj, "OnDoubleClick");
        handler2.Invoke(obj, new EventArgs());
    }
}
```

```
import System.*;

/**@delegate*/
delegate void MyDelegate(int i);

class Program
{
    public static void main(System.String [] args)
    {
        TakesADelegate(new MyDelegate(Program.DelegateFunction));
    }

    public static void TakesADelegate(MyDelegate SomeFunction)
    {
        SomeFunction.Invoke(21);
    }

    public static void DelegateFunction(int i)
    {
    Console.WriteLine("Called by delegate with number: " + i + ".");
   }
}
```

出力

```
test
test
```

Visual J# では、.NET Framework デリゲートの定義がサポートされています。Visual J++ 6.0 スタイルのデリゲートも定義できます。このデリゲートの型は、**com.ms.lang.Delegate** です。

デリゲート型の宣言は、次のコンポーネントで構成されます。

- /** @delegate */ 属性
- アクセス レベル
- キーワード デリゲート、またはマルチキャスト デリゲート
- デリゲート型が処理するメソッドの戻り値の型とシグネチャ
- デリゲート型の名前 (メソッドの戻り値の型とシグネチャとの間に記述)

EventHandler をパブリック デリゲート型として宣言する例を次に示します。このデリゲート型は、戻り値の型が void でパラメータを持たないメソッドを処理します。

```
/** @delegate */
public delegate void EventHandler();
```

一度に 1 つのメソッドだけを処理するために使用されるデリゲート型では、任意の戻り値の型とシグネチャのメンバ関数を処理できます。ただし、デリゲート型が同時に複数のメソッドを処理する場合は、戻り値の型を void にする必要があります。

.NET Framework デリゲートのバインディングは、常にコンパイル時に行われます。

出力

```
Called by delegate with number: 21.
```

参照

関連項目

.NET Framework を対象とするための構文

プロパティとイベントの公開

値型の使用

Java 言語では、**byte**、**int**、**boolean**、**char**、**short**、**long**、**float**、および **double** などのプリミティブ型は、値型です。他の型はすべて参照型です。他の .NET Framework 言語では、上記以外の値型を定義できます。Visual J# では、Java 言語が拡張されており、他の .NET Framework 言語で定義したこれらの値型を J# コード内で使用でき、Visual J# 2005 では、J# 言語内でも値型を作成できます。

値型は、参照型とはいくつかの点で異なります。以下の点は、値型に当てはまります。

- 値型はヒープではなくスタックに格納されます。
- 値型は、参照渡しではなく、値渡しされます。
- 値型を代入するとオブジェクト全体がコピーされます。参照のみがコピーされるわけではありません。
- 比較演算の場合には、値型オブジェクト全体の一致が比較されます。参照のみが比較されるわけではありません。
- 値型を派生させることはできません。
- 値型はコンストラクタを必要としません。

詳細については、「共通型システムの値型」を参照してください。

Visual J# コンパイラでは、.NET Framework またはその他のユーザー定義アセンブリで定義された値型を使用できます。値型のインスタンスは、Object として Visual J# で直接使用できます。Visual J# コンパイラでは、必要なボックス化が暗黙的に行われます。値型を **Object** 型の変数に代入すると、プロキシ オブジェクトが作成され、値型がその中にラップ (ボックス化) されます。同様に、オブジェクトを値型にキャストした場合には、コンパイラによってボックス化が解除され、オブジェクト ラップから値型が抽出されます。J# や他の言語で定義されている値型に関してはボックス化が自動的に行われますが、Java 言語のプリミティブ型については行われません。詳細については、「プリミティブ型に対応する値型の使用」および「ユーザー定義の値型」を参照してください。

使用例

```
// vjc_valuetypes1.jsl
import System.*;
public class MyClass
{
    public static void main(String [] args)
    {
        // DateTime is a value type; use it like a reference type.
        DateTime dt = new DateTime();
        dt = DateTime.Parse("01/01/2002 12:00");

        // Automatically box value type dt to System.Object.
        System.Object obj = dt;

        // Obj unboxed to the value type DateTime.
        DateTime dt2 = (DateTime) obj;
        dt2 = DateTime.Parse("01/01/2003 12:00");
    }
}
```

上のコードの `DateTime` は、.NET Framework ライブラリで定義された値型です。構文は参照型の場合と同じですが、値型での演算は、ヒープ ベースではなくスタック ベースで行われるため、生成されるコードは大きく異なります。値型は参照型に比べて、パフォーマンスが優れています。その理由は、ボックス化やボックス化解除の処理が必要ないこと、また、値型はヒープではなくスタックに格納されているため間接的にアクセスする必要がないことです。

参照

関連項目

.NET Framework を対象とするための構文
値型 (C# リファレンス)
プロパティとイベントの公開

プリミティブ型に対応する値型の使用

Visual J# では、.NET Framework の各プリミティブ型に対応するプリミティブ クラス型を直接使用できます (**int** の場合は **System.Int32**)。唯一の制限は、キャストを使って、対応するクラス型にプリミティブ型を代入する必要がある点です。また、**System.Object** にプリミティブ型を代入する場合にもキャストが必要です。これらの制限は、Java 言語のセマンティクスに違反しないようにするために必要です。プリミティブ型から対応する値型へのキャストは、推移的ではありません。

使用例

```
// vjc_valuetypes2.jsl
import System.*;
public class MyClass
{
   public static void main(String [] args)
   {
      Int32 int32 = (Int32) 10;
      int i = (int) int32;
      Console.WriteLine(i);
      System.Object obj1 = int32;   // Ok. int32 is a value type.
      Console.WriteLine(obj1);
      System.Object obj2 = (Int32) 10;   // Will also work.
      Console.WriteLine(obj2);

      // The following statements will not compile,
      // a primitive type cannot be assigned to a reference type:
      // System.Int32 int32 = 10;
      // System.Object obj = 10;
   }
}
```

出力

```
10
10
10
```

.NET Framework では、Java 言語の一部ではない各種のプリミティブ型をサポートしています。既に説明したように、.NET Framework から対応する値型を使うことにより、これらのプリミティブ型を使用できます。Visual J# コンパイラは、変換を考慮した正しいコードを生成します。次の表は、このような型の一覧です。

Visual J# でサポートされていない .NET Framework プリミティブ型	**.NET Framework の対応する値型**
unsigned short または **uint16**	**System.UInt16**
unsigned int または **uint32**	**System.UInt32**
unsigned long または **uint64**	**System.UInt64**
native int	**System.IntPtr**
native unsigned int	**System.UIntPtr**

参照

関連項目

.NET Framework を対象とするための構文

プロパティとイベントの公開

ユーザー定義の値型

値型はクラスに似ていますが、クラスはヒープ上のオブジェクトに対する参照であるのに対し、値型は、プリミティブ型同様、スタック上に直接作成できる集約型です。ただし、クラスと同じように、コンストラクタ、定数、フィールド、メソッド、プロパティ、およびメンバとして入れ子にされた型を持つことができます。一方、値型を定義する場合は、以下の制限が適用されます。

- 値型は、ValueType を拡張する必要があります。
- 値型は、明示的に **final** で宣言する必要があります。
- 値型を **abstract** で宣言することはできません。
- クラスのメンバは **public** や **private** で宣言できますが、**protected** では宣言できません。指定しない場合、アクセスは **package scoped** になります。
- 値型では、パラメータを持たないコンストラクタを宣言できません。
- 値型内のインスタンス フィールドには、フィールド初期化子を指定できません。ただし、静的フィールドの場合には、フィールド初期化子を指定できます。
- ユーザー定義の値型のコンストラクタで **super** を呼び出すとエラーになります。
- 値型では、**finalize** メソッドを宣言できません。
- 値型では、Java リフレクションがサポートされておらず、Java 型としてシリアル化できません。これらの型では、.NET Framework リフレクションおよび .NET Framework シリアル化を使用してください。
- 値型は、/x:net オプションと一緒に使用できません。

値型は 2 とおりの方法でインスタンス化できます。値型は、**new** 演算子を使用して、初期化されていない形式または初期化された形式でインスタンス化できます。**new** 演算子が呼び出されますが、値型オブジェクトは、ヒープ上ではなく、スタック上に割り当てられます。オブジェクトの有効期間は、そのオブジェクトが作成されるスタック フレームの有効期間と同じです。制御がスコープの外に出ると、そのスコープで宣言されているオブジェクトは破棄されます。

単純な値型を定義およびインスタンス化する例を次に示します。

使用例

```
// value_type.jsl
public final class Point extends System.ValueType
{
    public int x;
    public int y;
}
class CMain
{
    public static void main()
    {
        Point p = new Point();
        p.x = 5;
        p.y = 10;
        System.Console.WriteLine("Point [x,y] = " + p.x + ", " + p.y);
    }
}
```

出力

```
Point [x,y] = 5, 10
```

何も代入されていない値型、または **new** 演算子で初期化されていない値型は、初期化前の状態と見なされます。

値型には既定値があり、値型の配列が作成される場合など、特定のコンテキストで使用されます。既定値は、すべての値型フィールドにそれぞれの既定値を設定し、すべての参照型フィールドに null を設定することによって作成されます。たとえば、プリミティブな整数型の既定値はゼロです。したがって、次のコードでは、配列内のすべてのオブジェクトについて Point の `x` メンバと `y` メンバにゼロが設定されます。

```
Point[] pa = new Point[100];
```

各値型には、上で説明した方法で既定値を設定するパラメータなしのコンストラクタが暗黙的に定義されているため、パラメータを持たないコン

ストラクタを宣言するとエラーになります。

値型では、**ValueType** から継承したメンバを参照できます。

ユーザー定義の値型は、参照としてではなく、値として扱われます。参照型の場合には、1 つのオブジェクトに対して 2 つの参照が使用されることがありますが、値型の場合には、個々のインスタンスがそれぞれ独自のデータのコピーを持ちます。値型を代入すると、そのデータがコピーされるのに対し、参照型を代入すると、その参照だけがコピーされます。したがって、参照型の場合には、1 つのインスタンスに対する演算によって、そのインスタンスを共有する 2 つの参照に影響を与えることができますが、値型でこれを行うことはできず、演算の対象は常に、独自の一意なインスタンスとなります。

同様に、値型を関数の引数として渡した場合、関数で「参照渡しの引数を受け取るメソッドの定義」のタグを使用してパラメータを参照渡しすると指定されていない場合には、関数でオブジェクトの独自のコピーが作成されます。

参照

関連項目

値型の使用

.NET Framework を対象とするための構文

プロパティとイベントの公開

列挙型の使用

Visual J# コンパイラでは、.NET Framework またはその他のユーザー定義アセンブリで定義された列挙型を使用できます。特に指定しない限り、列挙型は参照型として扱われます。列挙型は、基になるプリミティブ型との間でキャストできます。また、Object に代入することもできます。この場合、コンパイラは必要なボックス化を暗黙に実行します。

使用例

```
import System.*;
public class MyClass
{
   public static void main()
   {
      DayOfWeek friday = DayOfWeek.Friday;

      // Call a method that has a parameter of type DayOfWeek
      TestClass mySchedule = new TestClass();
      mySchedule.SetNonWorkingDay(friday);
      // convert an enum to its underlying primitive:
      int i = (int) friday;
      Console.WriteLine(i);

      // OK to cast a primitive type to enum:
      DayOfWeek monday = (DayOfWeek) 1;
      Console.WriteLine(monday);

      // following line automatically boxes enum friday
      // to System.Object:
      System.Object obj = friday;
      Console.WriteLine(obj);
   }
}

public class TestClass
{
   public void SetNonWorkingDay(DayOfWeek dow)
   {
   }
}
```

出力

```
5
Monday
Friday
```

列挙型にはビット処理演算子 (|、&、^) を使用できます。`CheckBox.set_Anchor` の呼び出しで `System.Windows.Forms.AnchorStyles` に対してビットごとの OR 演算を実行するコードを次に示します。

```
CheckBox.set_Anchor(AnchorStyles.Top | AnchorStyles.Bottom)
```

参照

関連項目

enum キーワード
.NET Framework を対象とするための構文
プロパティとイベントの公開

ユーザー定義の列挙型 (J#)

Visual J# では、列挙型の作成がサポートされています。列挙型は **int** の一種で、その型で使用できる各種の値を表す名前付き定数を格納します。

列挙型は、ユーザー定義の値型です。値型のセマンティクスの規則が適用されます。詳細については、「ユーザー定義の値型」を参照してください。

列挙型の宣言は、クラスの宣言に似ています。ただし、それに加えて次の規則が適用されます。

- **class** キーワードの代わりに **enum** キーワードを使用します。
- 列挙型は、Enum を暗黙的に拡張します。
- 列挙型は、暗黙的に **static** かつ **final** です。
- 列挙型を **abstract** で宣言することはできません。
- 列挙型は、**public**、**private**、またはパッケージ スコープにできますが、**protected** にすることはできません。
- 列挙型は、メソッド、プロパティ、およびイベントを持つことができません。
- 列挙型は、インターフェイスを実装できません。
- 列挙型は /x (言語拡張機能の無効化) オプションと一緒に使用できません。

ある列挙型の名前付きの値は、その列挙型の **static final** メンバとして宣言され、定数式によって初期化されます。次に例を示します。

```
// enum1.jsl
// compile with: /target:library
public enum ColorEnum
{
    Red,
    Green,
    Blue
}
```

列挙値は、次のように、静的フィールドと同様に参照できます。

```
int i = (int) ColorEnum.Red
```

new キーワードを使用して列挙値をインスタンス化しようとすると、エラーになります。

enum メンバの整数値は、次の構文で設定できます。

```
public enum ColorEnum
{
    Red(1),
    Green(2),
    Blue(3)
}
```

明示的に設定しない場合、整数値は暗黙的に設定されます。列挙体の最初の列挙値にはゼロが設定されます。以降の各メンバの値は、明示的に設定しない場合、直前のメンバの値に 1 を加算することで決定されます。次に例を示します。

```
public enum ColorEnum
{
   Red,
   Green(10),
   Blue
}
```

この宣言例では、Red の値は 0、Green の値は 10、Blue の値は 11 になります。

列挙型の複数のメンバが、同じ整数値を共有することもできます。

enum のメンバ間では、依存関係が循環しない限り他のメンバとの依存関係が許可されます。

```
// enum2.jsl
// compile with: /target:library
public enum ColorEnum
{
   Red,
   Blue,
   Green(Blue);
}
```

Enum から継承したメソッドおよびプロパティは、列挙型の値で使用できます。

列挙型では、基底の型や他の整数型、または他の列挙型への暗黙の型変換は行われません。キャストを使用して明示的に変換する必要があります。次のコードは無効です。

```
int i = Color.Red; // error
int i = (int) Color.Red;  // OK
```

列挙型の値に対して使用できる演算子は、**==**、**!=**、**<**、**>**、**<=**、**>=**、**+**、**-**、**^**、**&**、**|**、および **~** です。

列挙型は値型であり、他の値型と同様にボックス化変換で使用できます。

列挙型は、switch ステートメントで使用できます。列挙体のメンバは、case ラベルとして使用できます。

```
ColorEnum color = Color.Red;
switch( color )
{
    case ColorEnum.Red:
    // ...
    break;
    case ColorEnum.Blue:
    // ...
    break;
    case ColorEnum.Green:
    //...
    break;
    default:
    break;
}
```

各 **enum** 型で定義される型は、それぞれ完全に異なる型です。2 つの **enum** 型の間で変換を行うには、明示的な列挙体変換が必要です。ある **enum** 型に保持できる値の種類は、その **enum** 型のメンバのみに限定されません。特に、ある **enum** 型の基になる型のすべての値は、その **enum** 型にキャストでき、その **enum** 型における固有値となります。

他の値型同様、列挙型では Java リフレクションがサポートされておらず、Java 型としてシリアル化できません。これらの型では、.NET Framework リフレクションおよび .NET Framework シリアル化を使用してください。

使用例

```
// enum_example.jsl
import System.*;

public enum RGBColor
{
    Red,
    Blue,
    Green
}

public final class RGB
{
    private int[] m_colors = new int[3];
```

```
    public RGB()
    {
        for (RGBColor rgb1 = RGBColor.Red;
            (int)rgb1 <= (int)RGBColor.Green;
            rgb1 = (RGBColor)((int)rgb1 + 1))
        {
        m_colors[(int)rgb1] = 0;
        }
    }

    public RGB(int r1, int r2, int r3)
    {
        // Enums must be converted to an integral type
        // for use as array indices.
    m_colors[(int)RGBColor.Red] = r1;
    m_colors[(int)RGBColor.Green] = r2;
    m_colors[(int)RGBColor.Blue] = r3;
    }

    int get_Color(RGBColor c)
    {
    return m_colors[(int)c];
    }

    public static void printRGBColor(RGBColor c)
    {
    switch( c )
    {
        case RGBColor.Red:
        Console.WriteLine("Red");
        break;
        case RGBColor.Blue:
             Console.WriteLine("Blue");
            break;
        case RGBColor.Green:
            Console.WriteLine("Green");
            break;
        default:
            break;
    }

    }

}

public class CMain
{
    public static void main()
    {
    RGB black = new RGB();
    RGB white = new RGB(255, 255, 255);

    Console.WriteLine("Black consists of R: " +
    black.get_Color(RGBColor.Red)
        + " G: " + black.get_Color(RGBColor.Green)
        + " B: " + black.get_Color(RGBColor.Blue));

    Console.WriteLine("White consists of R: " +
    white.get_Color(RGBColor.Red)
        + " G: " + white.get_Color(RGBColor.Green)
        + " B: " + white.get_Color(RGBColor.Blue));

    RGB.printRGBColor( RGBColor.Red);
  }
}
```

出力

```
Black consists of R: 0 G: 0 B: 0
White consists of R: 255 G: 255 B: 255
Red
```

参照

関連項目

列挙型の使用

enum キーワード

.NET Framework を対象とするための構文

プロパティとイベントの公開

プロパティとイベントの公開

プロパティとイベントを公開するコンポーネントを Visual J# で作成しておき、Visual Basic など別の言語でプログラムを記述して、このコンポーネントを使用することができます。

使用例

他の言語から使用可能なコンポーネントを作成するための J# のコードを次に示します。

```
// vjc_container.jsl
// compile with: /target:library
import System.Collections.ArrayList;

/** @delegate */
public delegate void ValueChanged();

public class Container
{
    private ArrayList al = new ArrayList();

    // Read-only property Count
    /** @property */
    public int get_Count()
    {
        return al.get_Count();
    }

    /** @property */
    public void set_Value(int i, System.String val)
    {
        al.Insert(i, val);
        if (ev != null)
        {
             ev.Invoke(); // Raise event.
        }
    }

    /** @property */
    public System.String get_Value(int i)
    {
        return (System.String) al.get_Item(i);
    }

    public ValueChanged ev = null;

    /** @event */
    public void add_Event(ValueChanged p)
    {
        ev = (ValueChanged) System.Delegate.Combine(ev, p);
    }

    /** @event */
    public void remove_Event(ValueChanged p)
    {
        ev = (ValueChanged) System.Delegate.Remove(ev, p);
    }
}
```

Visual J# コンポーネントのプロパティを使い、コンポーネントで発生するイベントをシンクする Visual Basic クライアントを次に示します。

```
' client.vb
' compile with: /reference:vjc_container.dll
Imports System
 Public Class Form
    WithEvents Dim c As New Container()
```

```
    Public Shared Sub Main()
        Dim f As New Form
    End Sub

    Public Sub New()
        c.Value(0) = "String1"
        c.Value(1) = "String2"

        ' Fetching read-only property
        System.Console.WriteLine(c.Count)
    End Sub

    ' Sink Visual J# event
    Public Sub EvListener() Handles c.Event
        System.Console.WriteLine("Visual J# Event Fired")
    End Sub
End Class
```

出力

```
Visual J# Event Fired
Visual J# Event Fired
2
```

参照

関連項目

.NET Framework を対象とするための構文
イベントの定義と使用
プロパティの定義と使用

ネイティブ メソッドの呼び出し

Visual J# は、Visual J++ 6.0 で使用できる J/Direct テクノロジを完全にサポートしています。また、共通言語ランタイムに用意されているプラットフォーム呼び出しサービスを使って、ネイティブ コードを呼び出すこともできます。詳細については、「アンマネージ DLL 関数の処理」を参照してください。

使用例

Microsoft Visual C++ ランタイム ライブラリ DLL を使ってテキストを出力するために、プラットフォーム呼び出しを利用するコードを次に示します。

```
// vjc_pinvoke.jsl
import System.Runtime.InteropServices.*;
class MyClass
{
    /** @attribute DllImport("msvcrt.dll") */
     public static native int puts(String c);
    /** @attribute DllImport("msvcrt.dll") */
     private static native int _flushall();

    public static void main(String [] args)
    {
        puts("Test");
        _flushall();
    }
}
```

出力

```
Test
```

プラットフォーム呼び出しを使ってネイティブ コードを呼び出すとき、マーシャリングがサポートされているのは、次の型に対してだけです。

- .NET Framework に定義されている型
- **java.lang.Object**
- **java.lang.String**
- プリミティブ型

その他の Java 言語型をネイティブ コードにマーシャリングするには、カスタム マーシャリングを使うことが必要な場合があります。これらの型は、.NET Framework プラットフォーム呼び出しの既定のマーシャラでは処理されないためです。詳細については、「相互運用マーシャリング」を参照してください。

Visual J++ 6.0 アプリケーションをアップグレードする場合、Visual J++ 6.0 の **@dll** ディレクティブや **@com** ディレクティブと .NET Framework 相互運用固有の属性を混在させることはできません。これらの属性は混在させないでください。これらの属性を混在させると、コンパイル後のアプリケーションで実行時に問題が発生する可能性があります。

参照

関連項目

.NET Framework を対象とするための構文

型変換演算

Visual J# では、型変換演算子がサポートされていません。通常、共通言語仕様 (CLS: Common Language Specification) 準拠の型には、変換を実行するための代替機構が用意されています。つまり、**ToX** (**X** は変換先の型名) メソッドおよび **FromY** (**Y** は変換元の型名) メソッドが提供されます。Visual J# のユーザーは、これらのメソッドを使って型変換を実行できます。

op_Implicit メソッドと **op_Explicit** メソッドは戻り値の型がオーバーライドされていることがあるため、メソッド解決には使用できません。したがって、この 2 つのメソッドを直接使うことはお勧めしません。

使用例

```
// vjc_type_conv_op.jsl
import System.*;
class MyClass
{
    public static void main(String [] args)
    {
        int i = 10;
        Decimal dec = new Decimal(20);
        Console.WriteLine("dec = {0}", dec);
        Console.Write("i = ");
        Console.WriteLine(i);

        // In Visual J#, you can use the CLS-compliant
        // conversion methods:
        i = Decimal.ToInt32(dec);
        Console.Write("i = ");
        Console.WriteLine(i);

        // Converting an int to a System.Decimal.
        // In Visual J#, you can use the constructor that
        // takes int:
        i = 9;
        dec = new System.Decimal(i);
        Console.WriteLine("dec = {0}", dec);
    }
}
```

出力

```
dec = 20
i = 10
i = 20
dec = 9
```

参照

関連項目

.NET Framework を対象とするための構文

.NET Framework 型と Java ラッパー型の変換

Visual J# 2005 の Java 言語ラッパー型は、**IConvertible** をサポートしているため、.NET ランタイム型との変換が可能です。

解説

IConvertible インターフェイスには、型を共通言語ランタイム型に変換できるメソッドが用意してあります。これらのメソッドの詳細については、IConvertible インターフェイスのドキュメントを参照してください。以下の Java 言語型に **IConvertible** が実装されています。

IConvertible をサポートしている型

java.lang.Boolean	**java.lang.Long**
java.lang.Byte	**java.lang.Short**
java.lang.Character	**java.lang.String**
java.lang.Double	**java.math.BigInteger**
java.lang.Float	**java.math.BigDecimal**
java.lang.Integer	**java.util.Date**

変換できない場合もあります。許可されていない変換を、**IConvertible** メソッドのいずれかを使用して行おうとすると、**InvalidCastException** がスローされます。以下の変換で例外がスローされます。

許可されていない変換

変換前	変換後
java.lang.Character	**System.Boolean**
java.lang.Character	**System.Single**
java.lang.Character	**System.Double**
java.lang.Character	**System.Decimal**
java.lang.Boolean	**System.Char**
java.lang.Float	**System.Char**
java.lang.Double	**System.Char**
java.math.BigDecimal	**System.Char**
java.math.BigInteger	**System.Char**
java.util.Date	**System.String** および **System.DateTime** を除くすべての型
java.lang.String および **java.util.Date** を除くすべての型	**System.DateTime**

参照

関連項目

IConvertible

InvalidCastException

.NET シリアル化のサポート

Visual J# 2005 では、J# クラス ライブラリ内に存在し Java シリアル化できる JDK クラスの .NET シリアル化がサポートされています。**Serializable** インターフェイスまたは **Externalizable** インターフェイスを使用してシリアル化できる J# の各 Java 言語型は、自動的に .NET Framework シリアル化できるようになりました。Visual J# に同梱の J# クラスは、.NET Framework API を使用するシリアル化に使用できます。

ユーザー定義クラスは、.NET Framework のシリアル化 API を使用してシリアル化するためには、すべて明示的に .NET Framework のシリアル化モデルに従う必要があります。

Java 言語でシリアル化できるこれらの型に対して、.NET Framework シリアル化が有効になることにより、次の動作が有効になります。

- シリアル化の際に **BinaryFormatter** または **SoapFormatter** を使用できます。
- これらの型のオブジェクトを .NET Framework リモート処理で使用できます。
- **XMLFormatter** の一部の機能をサポートできます。サポートが限定されるのは、.NET Framework シリアル化モデルではプライベート フィールドは通常シリアル化されないため、この場合にもプライベート フィールドがシリアル化されないからです。
- ユーザー定義クラスのメンバとして、これらの型を使用でき、メンバを .NET Framework シリアル化で使用できます。

カスタムのシリアル化処理を独自に記述することもできます。詳細については、「カスタムのシリアル化処理」を参照してください。

使用例

次の例では、**Serializable** を実装するオブジェクトを作成し、フィールドとして J# クラス ライブラリ型を使用します。また、**Serializable** 属性も明示的に割り当てることにより、Java 言語シリアル化モデルおよび .NET Framework シリアル化モデルの両方を使用してシリアル化できるようにしています。以前のバージョンでは、J# クラス ライブラリ型は .NET Framework でシリアル化できなかったため、この型は .NET Framework API でシリアル化できませんでした。オブジェクトは、両方の形式でシリアル化され、その後で再び読み込まれます。

```
// Serialize.jsl
// Serialization Example

import java.io.*;
import System.IO.*;
import System.Runtime.Serialization.Formatters.Binary.*;
import System.Runtime.Serialization.*;
import System.SerializableAttribute;

// This class implements the Java language Serializable interface
// thus making it a serializable class
// The user explicitly marks this class to be .NET Framework serializable.
// using the System.Serializable attribute.
// The class uses J# class library types as fields.
/** @attribute Serializable() */
public class MyClass implements Serializable
{
    public Integer i, j, k;

    public MyClass()
    {
      i = j = k = new Integer(0);
    }

    public MyClass(int _i, int _j, int _k)
    {
      i = new Integer(_i);
      j = new Integer(_j);
      k = new Integer(_k);
    }

    public void Write()
    {
       System.Console.WriteLine("i: " + i + " j: " + j + " k: " + k);
    }
}
```

```
public class CMain
{

   {
       System.out.println("Static ctor for CMain");
   }

   public static void WriteObjectNET(Object obj)
   {
       IFormatter formatter = new BinaryFormatter();
       Stream stream = new FileStream("MyFile.bin", FileMode.Create,
           FileAccess.Write, FileShare.None);
       formatter.Serialize(stream, obj);
       stream.Close();
   }

   public static void WriteObjectJava(Object obj) throws
   java.io.IOException
   {
       FileOutputStream fos = new FileOutputStream("serialized_Object");
       ObjectOutputStream os = new ObjectOutputStream(fos);
       os.writeObject(obj);
       os.flush();
       os.close();
   }

   public static Object ReadObjectNET()
   {
       IFormatter formatter = new BinaryFormatter();
       Stream stream = new FileStream("MyFile.bin", FileMode.Open,
           FileAccess.Read, FileShare.None);
       Object o = formatter.Deserialize(stream);
       stream.Close();
       return o;
   }

   public static Object ReadObjectJava() throws java.io.IOException,
   ClassNotFoundException
   {
       FileInputStream fis = new FileInputStream("serialized_Object");
       ObjectInputStream ois = new ObjectInputStream(fis);
       Object o = ois.readObject();
       ois.close();
       return o;
   }

   public static void main()
   {
       MyClass obj1 = new MyClass(100, 200, 300);
       try
       {
          WriteObjectNET(obj1);
          WriteObjectJava(obj1);
          obj1 = (MyClass) ReadObjectNET();
          obj1.Write();
          obj1 = (MyClass) ReadObjectJava();
          obj1.Write();
       }
       catch(Exception e)
       {
          System.Console.WriteLine("Exception!");
          System.Console.WriteLine(e.get_Message());
       }

   }
}
```

出力

```
i: 100 j: 200 k: 300
i: 100 j: 200 k: 300
```

参照

関連項目

.NET Framework を対象とするための構文

共通言語仕様への準拠の検証

共通言語仕様 (CLS : Common Language Specification) は、言語の相互運用性を向上するための規則セットです。フレームワークを複数の言語で使用できるようにするためには、この規則に準拠している必要があります。特定の型を別の言語で使用するためには、その型は CLS に準拠している必要があります。CLS 準拠とは、すべての言語に共有されていない言語構成要素の使用を避けることを意味します。CLS 準拠でない場合、その型をすべての言語で使用できるとは限りません。

Visual J# 2005 では、型が CLS に準拠しているかどうかを検証できる機能がコンパイラに用意されています。CLS の詳細については、「共通言語仕様の概要」を参照してください。

型を他の言語でも使用する場合は **CLSCompliant** 属性を設定する必要があります。また、CLS 準拠の型を含んでいるアセンブリには、アセンブリ レベルの **CLSCompliant** 属性を設定する必要があります。コンパイラは、CLS 準拠とマークされているアセンブリ内にある、**CLSCompliant** 属性が設定されている各パブリック型を検証します。

```
// MyClass.jsl
// compile with: /target:library
// Assembly-level CLS Compliant attribute:
/** @assembly System.CLSCompliant(true) */
// Class-level CLS Compliant attribute:
/** @attribute System.CLSCompliant(true) */
public class MyClass
{
    public void method1()
    {
    }
    public void method2()
    {
    }
}
```

アセンブリ レベルの **CLSCompliant** 属性を設定していない状態で、クラスにおいて **CLSCompliant** 属性を使用しないようにしてください。アセンブリ レベルの属性を設定していない場合、クラスの CLS 準拠は検証されません。

CLSCompliant 属性は **false** 引数と一緒に使用できます。これは、CLS 準拠の設定が行われているアセンブリ内の CLS に準拠していないクラスで必要です。クラスを `CLSComplaint(false)` とマークすると、このクラスは別の言語で使用不可のため、CLS 準拠の検証が不要なことを示します。

アセンブリ レベルの属性が 'false' に設定してある場合、CLSCompliant 属性が明示的に 'true' に設定してあるアセンブリ内の型のみを対象に CLS 準拠が検証されます。

パブリック型のみに `CLSCompliant(true)` でマークする必要があります。内部型は他の言語で使用できないため、内部型には CLS 規則が適用されません。

Visual J# によって適用される CLS 規則の一覧については、「共通言語仕様」を参照してください。

参照

その他の技術情報

言語サポート

J# のジェネリック

このトピックでは、J# 言語でのジェネリック型およびジェネリック メソッドの使用について説明します。

.NET Framework プラットフォームの一部の言語では、ジェネリック型およびメソッドの作成がサポートされています。ジェネリック型は、ビルド時に型パラメータが指定されず、実行時に任意の型を指定できる型です。置き換えられる型は、ジェネリック型の定義で指定されている制約を満たす必要があります。

ジェネリック型では、タイプ セーフなコレクション クラスがサポートされています。Object を項目の型として受け取るコレクション クラスを使用するのではなく、単一の型のみを受け入れる型指定されたコレクション クラスを作成できます。型指定されたジェネリック コレクションの場合、**Object** にキャストする必要がなく、コレクションからオブジェクトを読み取る場合にダウンキャストする必要もありません。キャストが不要になるとパフォーマンスが向上します。

.NET Framework のタイプ ライブラリには、Visual J# プログラムで使用できる、便利なジェネリック コレクション クラスが定義されています。詳細については、「System.Collections.Generic」を参照してください。

J# では、ジェネリック コレクション クラスの作成はサポートされていませんが、Java 言語の構文を拡張し、他の .NET Framework 言語で定義されたジェネリック型およびジェネリック メソッドから特化した形式の使用をサポートしています。このセクションでは、このような型を J# プログラムで使用する方法について説明します。

J# でのジェネリックの使用

J# でジェネリック型を特化させるための基本構文は、他の言語で使用される構文と同じです。次のように、型引数を山かっこで囲みます。

```
Stack<int> myIntStack = new Stack<int>();

Stack<String> myStringStack = new Stack<String>();

Dictionary<int, String> myDict = new Dictionary<int, String>();
```

これらの型は、構築された型と呼ばれます。同じジェネリック型から構築されていても、引数の型が異なる型どうしは、まったく異なる型です。つまり、 `Stack<int>` と `Stack<String>` は同じ型ではありません。これらの型はタイプ セーフであり、`Stack<int>` 型のスタックに文字列を代入しようとするとコンパイル エラーになります。

型引数には、必要に応じて、参照型、プリミティブ型、または値型を指定できます。

例

この例では、C# で定義したジェネリックを J# で使用しています。C# コードは、コンパイル後のアセンブリでは不明な、型パラメータ `T` を定義します。J# コードでは、このジェネリックに対して型引数を提供し、これによって、不明な型パラメータすべてを実際の型で置き換えたコンストラクタを生成します。

```
// stack.cs
// compile with: /target:library
// a generic class is just a family of classes defined
// in terms of some type T. T will be a particular
// type when the generic is consumed
public class Stack<T>
{
    private T[] items;  // field types can use 禅･
    private int nitems;

    public Stack()      // ctor creates a Stack<T> for any T
    {
        nitems = 0;
        items = new T[50];
    }

    // Pop() returns a value of type T by loading
    // it from the array of type T[]
    public T Pop()
    {
        if (nitems == 0)
        {
            System.Console.WriteLine("Stack underflow");
            return items[0];
        }
        return items[--nitems];
```

```
    }

    // Push() writes a T into the array, possibly expanding.
    public void Push(T item)
    {
    if (items.Length == nitems)
    {
        T[] temp = items;
        items = new T[nitems*2];
        System.Array.Copy(temp, items, nitems);
    }
        items[nitems++] = item;
    }
}
```

次の J# コードでは、上の C# コードで宣言したジェネリックを使用します。

```
// stackapp.jsl
// compile with: /reference:stack.dll
public class app
{
    public static void main(String [] args)
    {
        Stack<int> s = new Stack<int>();
        int i = 0;

        for (i = 0; i < 5; i++)
        {
            s.Push(i);
        }

        for (i = 5; i > 0; i--)
        {
            System.Console.WriteLine(s.Pop());
        }
    }
}
```

出力

```
4
3
2
1
0
```

例

インターフェイスもジェネリックにできます。ジェネリック インターフェイスには、ジェネリック クラス同様、不明な型パラメータがあります。

次の例は、ジェネリック インターフェイスを使用する J# コードを示します。

```
// istack.cs
// compile with: /target:library

public interface IStack<T>
{
    T Pop();
    void Push(T item);
}

public class Stack<T> : IStack<T>
{
    private T[] items;
    private int nitems;

    public Stack()
```

```
    {
        nitems = 0;
        items = new T[50];
    }

    public T Pop()
    {
        if (nitems == 0)
        {
            System.Console.WriteLine("Stack underflow");
            return items[0];
        }
        return items[--nitems];
    }

    public void Push(T item)
    {
        if (items.Length == nitems)
        {
            T[] temp = items;
            items = new T[nitems*2];
            System.Array.Copy(temp, items, nitems);
        }
        items[nitems++] = item;
    }
}
```

次の J# コードでは、上で宣言した C# のジェネリック クラスおよびジェネリック インターフェイスを使用します。

```
// istackapp.jsl
// compile with: /r:istack.dll
public class app
{
    public static void main(String [] args)
    {
        IStack<int> s = new Stack<int>();
        int i = 0;

        for (i = 0; i < 5; i++)
        {
            s.Push(i);
        }
        display(s);
    }

    // Note how the contructed type is passed as a param
    public static void display(
    IStack<int> s)
    {
        for (int i = 5; i > 0; i--)
        {
            System.Console.WriteLine(s.Pop());
        }
    }
}
```

出力

```
4
3
2
1
0
```

制約

.NET Framework クラス ライブラリまたは他の .NET Framework 言語で宣言したジェネリック型では、指定の型パラメータに対する型引数とし

て使用できる型を制限できます。たとえば、制約によって、特定のインターフェイスを実装する型や特定の基本クラスを継承する型に対して指定できる引数を制限できます。制約は通常、ジェネリック型の実装において、基本クラスやインターフェイスの一部のメソッドを使用する必要がある場合に行います。インターフェイスの制約および基本クラスの制約に加え、パラメータのないコンストラクタの制約など、特殊な制約も定義されています。パラメータなしのコンストラクタの制約では、型引数には既定のコンストラクタ (パブリックであり、パラメータのないコンストラクタ) が必要です。コンストラクタ制約を使用すると、ジェネリック型でその型のオブジェクトを確実に作成できます。

ジェネリック型に適用できる他の制約として、参照型制約があります。この制約では、型引数として使用する型は参照型である必要があります。反対に、使用する型を値型に制約することもできます。値型は、**int** や **double** などの組み込み型か、ユーザー定義の値型のいずれかです。

型パラメータに型を指定するたびに制約がチェックされます。たとえば、ジェネリック メソッドに対するメソッド呼び出しのときや、ジェネリック型から構築された型をビルドするときに制約がチェックされます。J# でジェネリック型に基づいて構築された型の場合、その構築された型を作成するときに制約がチェックされます。型引数 `T` に対する制約 `C` は、`T` から `C` への暗黙の変換が存在する場合に満たされます。

例

次の例は、型パラメータにインターフェイス制約を指定して C# で定義されたジェネリックを示しています。J# では、制約で指定されたインターフェイスを実装している型のみが、型引数として使用できます。

```
// idisplaystack.cs
// compile with: /target:library
public interface IDisplay
{
    void display();
}

public class Stack<T> where T : IDisplay
{
    private T[] items;
    private int nitems;

    public Stack()
    {
        nitems = 0;
        items = new T[50];
    }

    public T Pop()
    {
        if (nitems == 0)
        {
            System.Console.WriteLine("Stack underflow");
            return items[0];
        }
        return items[--nitems];
    }

    public void Push(T item)
    {
        if (items.Length == nitems)
        {
            T[] temp = items;
            items = new T[nitems*2];
            System.Array.Copy(temp, items, nitems);
        }
        items[nitems++] = item;
    }

    public void displayAll()
    {
        for (int i = 0; i < nitems; i++)
        {
           items[i].display();
        }
    }
}
```

次の J# コードでは、上で宣言した制約のある C# のジェネリック クラスを使用します。

```
// bookapp.jsl
// compile with: /reference:idisplaystack.dll
public class Book implements IDisplay
{
    String title;

    public Book(String s)
    {
        title = s;
    }

    public void display()
    {
        System.out.println(title);
    }
}

public class app
{
    public static void main(String [] args)
    {
        Stack<Book> s = new Stack<Book>();

        Book b1 = new Book("Alice in Wonderland");
        Book b2 = new Book("The Iliad");
        Book b3 = new Book("Paradise Lost");
        Book b4 = new Book("Robinson Crusoe");

        s.Push(b1);
        s.Push(b2);
        s.Push(b3);
        s.Push(b4);

        s.displayAll();
    }
}
```

出力

```
Alice in Wonderland
The Iliad
Paradise Lost
Robinson Crusoe
```

式

構築された型は、J# の式の一部として、他の型を使用できる場所で使用できます。たとえば、配列を作成する式で使用できます。

```
new Vector<String>[10];
```

キャストの型は、パラメータ化された型にできます。

たとえば、次のように宣言されているものとします。

```
class Dictionary<A, B> extends object { /* ... */ }

class Hashtable<A, B> extends Dictionary<A, B> { /* ... */ }

Dictionary<String, Integer> d;
```

次のキャストは有効です。正しい型引数による派生クラスへのキャストであるためです。

```
(Hashtable<String, Integer>) d;
```

一方、次の例は無効です。間違った型引数による派生クラスへのキャストであるためです。

```
(Hashtable<float, double>) d;
```

構築された型のインスタンスに対する **instanceof** 演算子は、その構築された型がインスタンスの型引数すべてと一致する場合、true を返します。

```
Stack<Integer> b = new Stack<Integer>();

return b instanceof Stack<Integer>; // should return true
```

一方、次のコードは決して true にならないため、コンパイル時のエラーになります。

```
return b instanceof Stack<String>;
```

構築された型へのアクセシビリティ

構築された型は、その構築された型を構成するすべての型がアクセス可能である場合にアクセスできます。これには、ジェネリック型自体の他に、型引数も含みます。ジェネリック型 `G` がパブリックであり、型引数 `T1` および `T2` がパブリックである場合、`G<T1, T2>` はパブリックです。`T1` または `T2` がプライベートであり、他がパブリックである場合、`G<T1, T2>` はプライベートです。構築された型のアクセシビリティは、その構成要素のうち最もアクセシビリティの低い要素と同じです。

ジェネリック メソッド

ジェネリック型とは異なり、ジェネリック メソッドには型引数を常に指定する必要はありません。明示的な型引数を指定せずにジェネリック メソッドを使用した場合、コンパイラは、そのメソッドに指定された引数から正しい型引数を推論します。あいまいさがある場合、コンパイラはエラーを出力します。必要である場合や、明確さが求められる場合は、引数を明示的に指定できます。型推論では、暗黙な変換および強制的な変換が無視されます。推論したパラメータに一貫性がない場合、たとえば、最初のメソッド パラメータで示唆されている型が 2 番目のメソッド パラメータでの型と異なる場合、型推論は失敗します。オーバーロードの解決は、型推論の後に行われます。

例

次の例では、ジェネリック配列を出力するために使用される printer.dll を作成します。

```
// printer.cs
// compile with: /t:library

public class Printer
{
    public void print<T>(T [] vals)
    {
        int lowerBound = vals.GetLowerBound(0);
        int upperBound = vals.GetUpperBound(0);

        for (int i = lowerBound; i <= upperBound; i++)
        {
            System.Console.WriteLine(vals[i]);
        }
    }
}
```

次の J# コードでは、上で宣言した C# のジェネリック ライブラリを使用します。

```
// printerapp.jsl
// compile with: /t:exe /reference:printer.dll
public class app
{
    public static void main(String [] args)
    {
        int [] ai = new int [2];
        ai[0] = 5;
        ai[1] = 6;

        Printer p = new Printer();
        p.print(ai);

        String [] books = new String [2];
        books[0] = "Alice in Wonderland";
        books[1] = "The Iliad";
        p.print(books);
    }
}
```

出力

```
5
6
Alice in Wonderland
The Iliad
```

汎用デリゲート

他の .NET 言語または .NET Framework で定義された汎用デリゲート型は J# コードで使用できます。デリゲートは、J# またはその他の .NET 言語にある型互換の任意の関数に代入できます。他のジェネリック型同様、汎用デリゲートは J# で作成できません。

例

C# で定義された Proc 汎用デリゲート型を J# アプリケーションで使用する例を次に示します。

```
// theDelegate.cs
// compile with: /target:library
public delegate void Proc<T>(T i);
```

次の J# コードでは、上の C# コードで定義した汎用デリゲート型を使用します。

```
// delegateapp.jsl
// compile with: /reference:theDelegate.dll

import java.util.Date;

public class MyDateClass
{
    public void DisplayDate(int i)
    {
        Date dt = new Date();
        System.out.println("The date is: " + dt.getDate());
        System.out.println("The value is: " + i);
    }
}

public class MyTimeClass
{
    public void DisplayTime(int i)
    {
        Date dt = new Date();
        System.out.println("The time is: " + dt.getHours() + 'h' + ' ' + dt.getMinutes() +
'm');
        System.out.println("The value is: " + i);
    }
}

public class app
{
    public static void main(String [] args)
    {
        MyDateClass d = new MyDateClass();
        Proc<int> p1 = new Proc<int>(d.DisplayDate);
        p1.Invoke(5);

        MyTimeClass    t = new MyTimeClass();
        p1 = new Proc<int>(t.DisplayTime);
        p1.Invoke(5);
    }
}
```

出力例

```
The date is: 10
The value is: 5
```

```
The time is: 10h 48m
The value is: 5
```

参照

関連項目

.NET Framework を対象とするための構文

その他の技術情報

Visual J# リファレンス

言語サポート

サポートされていない .NET Framework 機能

次に示す機能は、.NET Framework 共通言語ランタイムを使う他の言語ではサポートされている場合がありますが、Visual J# ではサポートされていません。

- 演算子のオーバーロードと、演算子のオーバーロードに関連付けられている .NET Framework セマンティクス
- **op_Implicit** および **op_Explicit** 変換演算子による、暗黙および明示的な型の変換
- Java 言語のデータ型と .NET Framework のデータ型の間でのシームレスな強制変換
- 同じ名前かつ同じシグネチャのメソッドを定義する必要のある 2 つのインターフェイスを単一の型に実装する場合、Visual J# では、これらのメソッドが区別されないため、別の実装を行う必要があります。Visual J# では、インターフェイス メソッドの名前とシグネチャが同じ場合には、記述できるコード本体も 1 つだけです。

参照

関連項目

サポートされていないクラス ライブラリと機能

Visual J# でサポートされていないシナリオ

その他の技術情報

言語サポート

チュートリアル : Java 言語でのソース コードの動的な生成とコンパイル

Visual J# には、Java 言語用の CodeDOM の実装が用意されています。VJSharpCodeProvider.dll アセンブリの Microsoft.VJSharp.VJSharpCodeProvider クラスには、Java 言語の ICodeGenerator インターフェイスと ICodeCompiler インターフェイスの Visual J# 実装のインスタンスを取得するためのメソッドが用意されています。

メモ :

使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

プロジェクトでの Visual J# コード プロバイダの指定

プロジェクトに Visual J# コード プロバイダを含めるには

- ソリューション エクスプローラでプロジェクトを右クリックし、[参照の追加] をポイントし、.NET Framework コンポーネントの一覧の [VJSharpCodeProvider] をクリックします。

CodeDOM および throws 句

CodeDOM 要素を使うと、言語に依存しない方法でソース コードをモデル化できます。**ICodeGenerator** および **ICodeCompiler** の言語固有の実装を使うと、コードを動的に生成およびコンパイルできます。

CodeDOM の詳細については、「複数の言語でのソース コードの動的な生成とコンパイル」を参照してください。

CodeDOM は、共通言語ランタイム プログラミング言語に含まれている共通の型のコード要素をサポートします。ただし、プログラミング言語のすべての機能を表す要素を用意するように設計されているわけではありません。

throws 句、**extends** と **implements** の区別、値型を含むキャストの指定など、Visual J# 固有の機能をサポートする方法について説明します。

メモ :

他の言語による **ICodeGenerator** の実装では、同じ CodeDOM オブジェクト グラフを渡しても、その言語に関係のない UserData 情報は無視されます。

CodeDOM には、メソッドの **throws** 句で宣言された例外を表す方法はありません。

throws 句を表すには

1. CodeMemberMethod オブジェクトの CodeObject.UserData メンバに、**throws** 句が指定されたメソッドを表すキーと値の組み合わせを設定します。

   このメンバは、**ICodeGenerator** の Visual J# 実装で正しく解釈されます。

2. このキーに **throwsCollection** という名前を付けます。値には、メソッドによってスローされる例外を記述する CodeTypeReference オブジェクトのコレクションを表す CodeTypeReferenceCollection オブジェクトを指定します。

3. Visual J# コード ジェネレータは、**throwsCollection** を使ってメソッドの **throws** 句を生成します。

   詳細については、「CodeDOM サンプル (CodeDomProvider の使用)」を参照してください。

クラス宣言の生成

CodeDOM には、基本インターフェイスと区別してクラスの基本クラスを表す方法はありません。既定では、Visual J# コード ジェネレータは、CodeTypeDeclaration.BaseTypes を次のように解釈します。

- BaseTypes コレクションの最初の型は、そのクラスの基本クラスであり、**extends** 句に対応します。
- **BaseTypes** コレクションのその他の型は、そのクラスによって実装されるインターフェイスであり、**implements** 句に対応します。

明示的な **extends** 句を含まないクラス宣言を生成するには、Visual J# コード ジェネレータに追加情報が必要になります。

extends 句と implements 句を区別するには

1. CodeTypeDeclaration オブジェクトの **UserData** メンバに、生成するクラスを表すキーと値の組み合わせを設定します。
2. このキーに **hasExtendsClause** という名前を付けます。値には、**Boolean** オブジェクトを指定します。
3. コード ジェネレータは、この値を次のように解釈します。
   - **hasExtendsClause** が true の場合、**BaseTypes** コレクションの最初の型は、そのクラスの基本クラスであり、**extends** 句に対応します。**BaseTypes** コレクションのその他の型は、そのクラスが **implements** 句を使用して実装するインターフェイスです。
   - **hasExtendsClause** が false の場合、**BaseTypes** コレクションのすべての型は、そのクラスによって実装されるインターフェイスとして解釈されます。

詳細については、「CodeDOM サンプル (CodeDomProvider の使用)」を参照してください。

J# では、自動的なボックス化およびボックス化解除がサポートされていないため、.NET の他の言語でのキャストよりも明示的なキャストが必要です。J# では、int に対して **System.Int32** を使用するなど、適切なラッパー型にキャストすることによって、明示的なボックス化を指定する必要があります。J# では、ダブル キャストを使用して、キャストを複数回行うことが必要になる場合があります。たとえば、浮動小数点値からボックス化された **System.Int32** にキャストする場合は、**int** にキャストしてから、**System.Int32** にキャストして、その整数をボックス化します。J# の CodeCastExpression オブジェクトでは、その UserData メンバを使用して、必要なキャスト型を正確に指定できるようになりました。**UserData** メンバには、**Boolean** 型の 2 つのキーがあり、使用可能なキャストの種類を指定できます。

値型に関連するキャストを指定するには

1. **CodeCastExpression** オブジェクトの **UserData** メンバに、生成するキャスト式を表す 2 つのキーの組み合わせを設定します。
2. キーの一方に **CastIsBoxing** という名前を付け、もう一方に **DoubleCast** という名前を付けます。
3. これらのキーの値として **Boolean** オブジェクトを代入します。この **Boolean** 変数の値により、次の表に従って、生成されるキャストの種類が決定します。

CastIsBoxing	DoubleCast	生成されるキャスト
false	true	(int) (System.Int32)
true	false	(System.Int32)
true	true	(System.Int32)(int)
false	false	(int)

実際の型は、**CodeCastExpression** オブジェクトの TargetType プロパティで指定された型になります。ここでは、単に説明のために **int** 型と **System.Int32** 型を使用しています。

参照

処理手順

CodeDOM サンプル (CodeDomProvider の使用)

その他の技術情報

言語サポート

動的なソース コードの生成とコンパイル

Visual J# のドキュメント コメント

ここでは、Visual J# 開発環境でサポートされているドキュメント コメントについて説明します。

このセクションの内容

Visual J# コード エディタでのドキュメント コメントのサポート

ドキュメント コメントをサポートするコード エディタの機能について説明します。ドキュメント コメントの形式は、Java 言語ソース コードで Javadoc コメントと呼ばれているコメント形式です。

Javadoc コメントと等価な XML ドキュメント

[Web ページのビルド コメント] コマンドについて説明します。このメニュー コマンドを使うと、ソース ファイルからドキュメントを生成できます。

方法 : Javadoc コメントから XML ドキュメントを生成する

Javadoc コメントから .NET Framework の標準ドキュメント コメントを生成する方法について説明します。

関連するセクション

言語サポート

Visual J++® 6.0 と .NET Framework の、Visual J# でサポートされている機能とサポートされていない機能に関するリファレンスです。

Visual J# コード エディタでのドキュメント コメントのサポート

Visual J# では、Java 言語ソース コードで Javadoc コメントと呼ばれている形式のドキュメント コメントがサポートされています。

ドキュメント コメントのオートコンプリート

Visual J# コード エディタでは、Javadoc コメントの開始文字 /** を入力すると、次の行にコメントの終了文字 */ が自動的に挿入されます。

Javadoc コメントでのタグのオートコンプリート

Javadoc コメント内で、Javadoc タグと Visual J++ 6.0 の Microsoft 拡張ディレクティブは、どちらも @ 文字で始まります。Visual J# では、Javadoc コメント内で @ 文字を入力すると、サポートされている Javadoc タグと Microsoft 拡張ディレクティブの両方を含むオートコンプリート リストが表示されます。

Javadoc コメント内での IntelliSense

Visual J# プログラムでは、属性宣言構文を Javadoc コメント形式で使うことができます。Javadoc コメント内で「**@attribute**」と入力すると、アクセスできるクラスの一覧が、IntelliSense ドロップダウン リストに表示されます。

参照

その他の技術情報

Visual J# のドキュメント コメント

方法：Javadoc コメントから XML ドキュメントを生成する

ファイルまたはプロジェクト内のコメントを、Java 言語のコメント形式から .NET 標準のコメント形式に変換する方法について説明します。

メモ：
使用している設定またはエディションによっては、表示されるダイアログ ボックスやメニュー コマンドがヘルプに記載されている内容と異なる場合があります。設定を変更するには、[ツール] メニューの [設定のインポートとエクスポート] をクリックします。詳細については、「Visual Studio の設定」を参照してください。

Javadoc コメントから XML コメントを生成するには

1. Visual J# プロジェクト内で、Javadoc コメントのあるプロジェクト ノードを右クリックします。
2. ショートカット メニューの [プロパティ] をクリックします。
3. [プロジェクトの設定] ウィンドウで左側にある [ビルド] をクリックして、ビルド ペインを開きます。
4. [XML ドキュメント ファイル] チェック ボックスをオンにします。
5. テキスト ボックスに、XML ドキュメント ファイルの名前を入力します。
6. [プロジェクトの設定] ウィンドウを閉じます。
7. [ビルド] メニューの [ソリューションのビルド] または [<projectname> のビルド] をクリックします。ファイル内の Javadoc コメントに基づいて、このプロジェクト用の XML ドキュメント ファイルが作成されます。

参照

関連項目

Javadoc コメントと等価な XML ドキュメント

その他の技術情報

Visual J# のドキュメント コメント

Javadoc コメントと等価な XML ドキュメント

XML ドキュメントは Javadoc コメントから作成できます。ここでは、Javadoc コメントに等価な XML ドキュメントについて説明します。

解説

Javadoc コメントと等価な XML ドキュメントを次の表に示します。

Javadoc コメント タグ	**等価な XML ドキュメント**
なし	<remarks> または <summary>
@author *name*	<author> *name* </author>
@deprecated *description*	<obsolete> *description* </obsolete>
@exception *exception-class text*	<exception cref="*exception-class*">*text*</exception>
@param *paramname text*	<param name=*paramname*> *text* </param>
@return *text*	<returns>*text*</returns>
@see *typename*	<seealso cref="*typename*"/>
@see *typename#member*	<seealso cref="*typename.member*"/>
@value *property-description*	<value> *property-description* </value>
@version *text*	<version> *text* </version>

メモ :

@value タグは Visual J# 固有です。

メモ :

タグのないコメントは、型の場合には <remarks> タグに変換され、メソッド、フィールド、またはコンストラクタの場合には <summary> タグに変換されます。

参照

処理手順

方法 : Javadoc コメントから XML ドキュメントを生成する

Visual J# の例外のサポート

このセクションでは、JDK の仕様で指定されている J# 例外の階層構造と .NET Framework 例外の階層構造、およびそれらの相互関係について説明します。

このセクションの内容

Visual J# の例外の階層構造

Visual J# と .NET Framework の例外、およびそれらの相互関係について説明します。

Visual J# のデバッガ用例外処理

デバッガ用の例外処理の設定について説明し、例外処理を設定する必要がある理由について説明します。

Visual J# 例外

J# の例外と、それに対応する .NET Framework の例外の一覧です。

参照

その他の技術情報

Visual Studio でのデバッグ
言語サポート

Visual J# の例外の階層構造

Visual J# は、Java 言語例外および .NET Framework 例外の 2 種類のエラー メッセージを生成します。

- Java 言語例外は **java.lang.Throwable** から派生します。これらは次の 2 つのカテゴリに分かれます。

例外	説明
checked	メソッドを呼び出して checked 例外がスローされると、メソッドを呼び出したコードに checked 例外をキャッチする `try-catch` ブロックが含まれていない限り、コンパイラはエラーを表示します。
Runtime	**java.lang.RuntimeException** から派生します。コンパイラは呼び出し元のコードによるキャッチを強制しません。

- .NET Framework 例外は Exception から派生します。これらの例外には、前述のような分類はありません。

Java 言語のランタイム例外の多くに対しては、同等の意味を持つ .NET Framework 例外があります。たとえば、**java.lang.NullPointerException** は、NullReferenceException に似ています。Java 言語のランタイム例外と .NET Framework 例外のマップについては、「Visual J# 例外」の表を参照してください。

ソース コード内で Java 言語のランタイム例外をキャッチする場合は、生成される実行可能コードでその Java 言語のランタイム例外に対応する .NET Framework 例外もキャッチすることが、Visual J# コンパイラによって保証されます。次に例を示します。

```
try
{
    methodWithNullReference();
}
catch (java.lang.NullPointerException e)
{
    // Catches both java.lang.NullPointerException
    // and System.NullReferenceException.
    e.printStackTrace();
}
```

この例では、次の 2 つの場合に null 参照の例外が発生する可能性があります。

- `methodWithNullReference` で **NullPointerException** として明示的にスローされる場合

  または

- .NET Framework **NullReferenceException** として実際に実行時に発生する場合

この `catch (java.lang.NullPointerException e)` 句は、**NullPointerException** と **NullReferenceException** のいずれの例外もキャッチします。

Java 言語固有の例外をキャッチしている場合、対応する Java 言語ランタイム例外を持つ .NET Framework 例外は一般に無視できます。ただし、対応する Java 言語ランタイム例外を持たない .NET Framework 例外、つまり、「Visual J# 例外」の表にない例外については、**catch** 句で指定する必要があります。

例外のマップについての変更点

Visual J# 2005 では、例外のマップ動作が変更されました。Java 言語で **java.lang.Throwable** をキャッチすることによって通常可能であるような、`catch all exceptions` を単一の **catch** 句にインクルードする方法は、以前は使用できませんでした。以前のバージョンでは、**java.lang.Throwable** をキャッチする **catch** 句では .NET Framework 例外がキャッチされませんでした。現在は、Visual J# コンパイラに変更が加えられ、**java.lang.Throwable** をキャッチする単一の **catch** 句ですべての例外をキャッチできるようになりました。必要に応じて、以前と同じように .NET Framework 例外を個別の **catch** 句でキャッチすることもできます。

次の例は、有効な一連の **catch** 句およびそれぞれの動作を示しています。**java.lang.Throwable** のキャッチに加えて、さまざまな .NET Framework 例外をキャッチする明示的な **catch** を含める場合、その動作は、前のセクションで定義したように、例外がマップされているかどうかによって異なります。マップされる例外は **java.lang.Throwable** の **catch** 句によってキャッチされ、マップされない例外は、その例外に特化した固有の **catch** 句によってキャッチされます。

例 1

```
// Example 1: Throwable catches all exceptions.
```

```
try
{
    // ...
}
catch (Throwable t)
{
    // ...
}
```

上の **catch** ステートメントは、すべての例外をキャッチします。次のコードは、生成される中間言語コードと同等な J# コードです。

```
try
{
    // ...
}
catch (System.Exception ex)
{
    // The original runtime filter is replaced with this
    // catch clause.
    Throwable t = Throwable.wrapException(ex);
    if (t == null) rethrow;
    // ...
}
catch (Throwable t)
{
    // ...
}
```

例 2

スタック フレーム全体を取得するには、ラップされた内部例外を **get_InnerException()** を呼び出して抽出し、その内部例外に対して **get_StackTrace()** を呼び出す必要があります。

```
// Example 2: Explicitly catching .NET Framework exceptions still works
// but we get more details with the get_StackTrace() method.
try
{
    // ...
}
catch (Throwable t)
{
  System.out.println(t.get_InnerException().get_StackTrace());
}
catch (System.Exception except)
{
    // ...
}
```

前述のケースで、**Exception** を明示的にキャッチする **catch** 句が存在する場合、**java.lang.Throwable** の catch 句では Java 言語例外のみがキャッチされます。マップは行われません。

例 3

```
// Example 3: Catching an unmapped .NET exception subclass.
try
{
    // ...
}
catch (Throwable t)
{
    // ...
}
catch (ExceptionUnmapped except)
{
    // ...
}
```

以前と同じように、ExceptionUnmapped に一致するすべての例外が **java.lang.Throwable** 句の catch 句ではなく ExceptionUnmapped **catch** 句の catch 句によってキャッチされます。次のコードは、マップの結果を示します。

```
try
{
    // ...
}
catch (ExceptionUnmapped except)
{
    // ...
}
catch (System.Exception ex)
{
    // Note the change in order of the catch clauses
    // so that binary behavior is maintained.
    Throwable t = Throwable.wrapException(ex);
    if (t == null)
    {
        rethrow;
    }
}
catch (Throwable t)
{
    // ...
}
```

例 4

```
// Example 4: Catching a mapped .NET Framework Exception subclass.
try
{
    // ...
}
catch (Throwable t)
{
    // ...
}
catch (ExceptionMapped except)
{
    // ...
}
```

前述の例の場合、マップされた例外を **java.lang.Throwable** の catch 句でキャッチしようとしていますが、この **catch** 句に到達できる例外がないというコンパイラのエラーが発生します。

デバッガでは、**java.lang.Throwable** と **Exception** は現在でも別の例外クラスとして扱われます。たとえば、デバッガの既定の動作を設定する場合は、これら 2 つの例外グループそれぞれに対して設定する必要があります。

参照

その他の技術情報

言語サポート

Visual J# のデバッガ用例外処理

ここでは、デバッガ用例外処理の設定について説明し、なぜ例外処理を設定する必要があるのかについて解説します。

メモ：
対応する .NET Framework での例外一覧を確認するには、「Visual J# 例外」を参照してください。

開発環境での例外処理の設定

[例外] ダイアログ ボックスの [Common Language Runtime Exceptions] に、[Java Language Exceptions] カテゴリがあります。このカテゴリに含まれるの例外の中には、対応する例外が .NET Framework 存在するものがあります。つまり、そのような例外で、Java 言語例外の処理を変更する場合には、対応する .NET Framework 例外の処理も変更して、Java 言語例外の動作と同じように動作させる必要があります。その理由は、例外は、Visual J# に付属するクラス ライブラリ、または基になる .NET Framework 共通言語ランタイム層から発生するためです。

例外処理の設定が必要な理由

例外処理コードが存在する場合、Visual J# コンパイラでは、適切な命令コードが出力され、.NET Framework 例外が対応するクラス ライブラリ例外にマップされます。Visual J# の例外と、スローされた対応する .NET Framework の例外の両方に対して、例外ハンドラ コードが実行されます。ただし、デバッガではこの変換が認識されません。そのため、キャッチされない例外で中断する場合や、例外が発生した場合は、クラス ライブラリの例外および対応する .NET Framework の例外の両方の処理を変更する必要があります。

使用例

Visual J# の次のコード例を参照してください。

```
public void caller()
{
   // Handler for Visual J# exception:
   try {
      stringize(null); // Will result in an exception at runtime:
   } catch (java.lang.NullPointerException nullEx) {
      // Compiler emits code
      // to catch System.NullReferenceException also.
      nullEx.printStackTrace();
   }
}

public String stringize(Object o) {
   // Invokes method without checking for null.
   // Results in a System.NullReferenceException at run time.
   return o.toString();
}
```

実行時にパラメータを *null* にして **stringize** メソッドを呼び出すと、NullReferenceException が発生します。一方、コンパイラにより、**java.lang.NullPointerException** と **NullReferenceException** の両方を処理するコードが出力されます。したがって、ユーザー コードでは透過的な方法で例外をキャッチできます。

一方、Visual Studio でデバッグしている場合、null 値を **stringize** メソッドに渡した際の例外処理を変更するには、処理を **java.lang.NullPointerException** に変更するだけでは不十分です。デバッガにより、実行時に **NullReferenceException** が発生します。したがって、この例外用にも処理を変更する必要があります。

stringize メソッドが別のクラス内に存在する場合には、**java.lang.NullPointerException** が明示的にスローされるため、上記 2 つの例外の処理を変更する必要があります。両方の状況に対応するには、両方の例外の総合処理を設定する必要があります。

参照

関連項目

Visual J# の例外の階層構造

概念

条件付きデバッグのサポート

Visual J# 例外

Visual J# アプリケーションの開発者は、次の 2 つの例外の階層構造を使用する必要があります。

- **java.lang.Throwable** から派生する Java 言語例外。これらの例外は次の 2 つのカテゴリに分かれます。
    - checked 例外

      メソッドを呼び出して checked 例外がスローされると、メソッドを呼び出したコードに checked 例外をキャッチする **try-catch** ブロックが含まれていない限り、コンパイラはエラーを表示します。
    - ランタイム例外

      java.lang.RuntimeException の派生クラス。コンパイラは呼び出し元のコードによるキャッチを強制しません。
- Exception から派生する .NET Framework 例外。

Java 言語のランタイム例外のほとんどに対しては、同等の意味を持つ .NET Framework 例外があります。たとえば、**java.lang.NullPointerException** は NullReferenceException に類似しています。

ソース コード内で Java 言語のランタイム例外をキャッチする場合は、生成される実行可能コードでその Java 言語のランタイム例外に対応する .NET Framework 例外もキャッチすることが、Visual J# コンパイラによって保証されます。

メモ :
処理されるランタイム例外は、デバッガとエラー コンソールの両方で J# 例外として表示されます (JDK 1.1.4 仕様に従います)。処理されないランタイム例外は、デバッガでは .NET 例外として表示され、エラー コンソールでは J# 例外として表示されます。

次の表は、Java 言語例外とそれに対応する .NET Framework 例外の一覧です。

Java 言語の例外	**.NET Framework の例外**
java.lang.ArithmeticException	ArithmeticException、DivideByZeroException、OverflowException 詳細については、以下のトピックを参照してください。 ● 例外のトラブルシューティング : System.DivideByZeroException ● 例外のトラブルシューティング : System.OverflowException
java.lang.ArrayIndexOutOfBoundsException	IndexOutOfRangeException 例外のトラブルシューティング : System.IndexOutOfRangeException を参照してください。
java.lang.ArrayStoreException	ArrayTypeMismatchException 例外のトラブルシューティング : System.ArrayTypeMismatchException を参照してください。
Java.lang.ClassCastException	例外のトラブルシューティング : System.InvalidCastException を参照してください。
java.lang.IllegalAccessError	FieldAccessException、MethodAccessException 詳細については、以下のトピックを参照してください。 ● 例外のトラブルシューティング : System.FieldAccessException ● 例外のトラブルシューティング : System.MethodAccessException

java.lang.IllegalArgumentException	ArgumentException、ArgumentNullException、ArgumentOutOfRangeException、 RankException 詳細については、以下のトピックを参照してください。 • 例外のトラブルシューティング : System.ArgumentNullException • 例外のトラブルシューティング : System.RankException.
java.lang.IncompatibleClassChangeError	MissingMemberException 例外のトラブルシューティング : System.MissingMemberException を参照してください。
java.lang.NoSuchFieldError	例外のトラブルシューティング : System.MissingFieldException を参照してください。
java.lang.NoSuchMethodError	例外のトラブルシューティング : System.MissingMethodException を参照してください。
java.lang.NumberFormatException	例外のトラブルシューティング : System.FormatException を参照してください。
java.lang.OutOfMemoryError	例外のトラブルシューティング : System.OutOfMemoryException を参照してください。
java.lang.SecurityException	例外のトラブルシューティング : System.Security.SecurityException を参照してください。
java.lang.StackOverflowError	例外のトラブルシューティング : System.StackOverflowException を参照してください。
java.lang.VerifyError	例外のトラブルシューティング : System.Security.VerificationException を参照してください。

参照

関連項目

Visual J# の例外の階層構造

Visual J# のデバッガ用例外処理

クラス ライブラリのサポート

Visual J# には、JDK レベル 1.1.4 のほとんどのクラス ライブラリと同等の機能をサポートする独自に開発されたクラス ライブラリのセットが用意されており、College Board の Advanced Placement カリキュラムの Computer Science で指定されているクラスが含まれています。また、Visual J# では、WFC (Windows Foundation Classes) やその他の多くの **com.ms.*** クラス ライブラリなど、Microsoft Visual J++® 6.0 に付属の Microsoft® 拡張機能もサポートされています。

このセクションでは、Visual J# でサポートされているパッケージとサポートされていないパッケージについて説明します。また、クラス ライブラリのサポートに関する実装固有の詳細情報についても説明します。

このセクションの内容

サポートされているクラス ライブラリ

Visual J# で使用できるクラス ライブラリの一覧です。

サポートされていないクラス ライブラリと機能

Visual J# でサポートされていないクラス ライブラリと API の一覧です。

追加のクラス ライブラリのサポート

JDK 1.2 レベルと同等の機能が実装されているクラスとインターフェイスの一覧です。

java.lang.Class クラスのメソッド

java.lang.Class の **Class.FromType** メソッドおよび **Class.ToType** メソッドの詳細です。

com.ms.wfc.app.Locale クラスのメソッド

使用を推奨されていない **com.mswfc.app.Locale** クラスのメソッドと、それに代わるメソッドの一覧です。

com.ms.vjsharp.text.FormatDefaults クラスのメソッド

Visual J# でサポートされている **com.ms.vjsharp.text.FormatDefaults** メソッドについて説明します。

内部例外

J# 例外クラスの動作について説明します。

Visual J++ 6.0 との互換性の問題

Java 言語および JDK レベル 1.1.4 のさまざまなパッケージについて、Visual J# と Visual J++ 6.0 との互換性の問題を詳しく説明します。

Supplemental UI Library

Supplemental UI Library を使用して、Microsoft Windows の外観と操作性を備えたアプリケーションを開発する方法について説明します。

ThreadDeath 互換性クラスの警告

J++ と J# におけるスレッドの動作の違いについて説明します。

関連するセクション

リファレンス

Visual J# コンパイラ、バイナリ変換ツール (JbImp.exe)、言語とライブラリのサポート、Visual J++ 6.0 プロジェクトのアップグレード、.NET Framework クラス ライブラリ用の Visual J# 構文、および開発環境に関するリファレンスです。

言語サポート

Visual J# でサポートされている、またはサポートされていない Visual J++ 6.0 および .NET Framework の言語機能に関する参照情報を示します。

サポートされているクラス ライブラリ

Visual J# は、Visual J++ 6.0 に含まれていた JDK レベル 1.1.4 のほとんどのパッケージと同等の機能を持つクラス ライブラリをサポートするように設計されています。JDK レベル 1.2 の機能に相当する機能をサポートしているクラスもあります。詳細については、「追加のクラス ライブラリのサポート」を参照してください。

Visual J# では、Microsoft Windows と同様な外観と操作性を持ったアプリケーションの開発がサポートされています。関連するクラスは **com.ms.vjsharp.swing.plaf.windows** パッケージに含まれています。詳細については、「Supplemental UI Library」を参照してください。

Visual J# には、WFC (Windows Foundation Classes) ライブラリをサポートするクラスの他に、大学入試センターのコンピュータ サイエンス用飛び級カリキュラムでの指定に対応したクラスや、以下に示した多数の com.ms.* パッケージが用意されています。

- **com.ms.lang**
- **com.ms.dll**
- **com.ms.com**
- **com.ms.win32**
- **com.ms.util**
- **com.ms.jdbc.odbc**
- **com.ms.vjsharp.swing.plaf.windows**

参照

関連項目

サポートされていないクラス ライブラリと機能

概念

Visual J# Class Library

その他の技術情報

クラス ライブラリのサポート

サポートされていないクラス ライブラリと機能

Visual J# では、以下の機能はサポートされていません。

- RMI (Remote Method Invocation)、JNI (Java Native Interface)、および RNI (Raw Native Interface) のテクノロジ。
- バイトコード (.class ファイル) からのクラスの読み込み。ただし、Microsoft Intermediate Language (MSIL) アセンブリからのクラスの読み込みはサポートされています。
- CLASSPATH 変数。実行時にクラスやリソースの検索と読み込みを行うための代替手段が用意されています。詳細については、「サポートされていない CLASSPATH 変数」を参照してください。

Visual J# では、次のクラス ライブラリおよび API を使用できません。

- 「サポートされているクラス ライブラリ」の一覧にある以外の **com.ms.*** パッケージ。サポートされていないパッケージのほとんどについて、アップグレードによる対処方法が、Visual J# のドキュメントで説明されています。詳細については、「Visual J++ 6.0 からのアップグレード」を参照してください。
- Visual J++ 6.0 に付属の JDK レベル 1.1.4 のクラス ライブラリに含まれていない、sun.*、netscape.* などのパッケージ。
- Visual J++ 6.0 で提供されていたプロファイラ、ヒープ監視、およびデバッグ API はサポートされていません。代わりに、.NET Framework ライブラリの一部として用意されている代替 API を使います。詳細については、「アプリケーションのデバッグとプロファイリング」を参照してください。

参照

関連項目

サポートされていない .NET Framework 機能
Visual J# でサポートされていないシナリオ
サポートされているクラス ライブラリ

その他の技術情報

クラス ライブラリのサポート

追加のクラス ライブラリのサポート

JDK 1.2 レベルのクラスに相当する機能をサポートするために、vjslib.dll のクラスおよびインターフェイスのサブセットが変更されています。**java.util** 内の全クラスに、JDK 1.2 レベルのクラスに相当する機能が実装されています。

JDK 1.2 の機能

以下のクラスおよびインターフェイスにも JDK 1.2 レベルに相当する機能が実装されています。

パッケージ	クラスまたはインターフェイス
java.io	**File**
java.io	**FileFilter**
java.lang	**Byte**
java.lang	**Character**
java.lang	**Double**
java.lang	**Float**
java.lang	**Integer**
java.lang	**Long**
java.lang	**Math**
java.lang	**Short**
java.lang	**StringBuffer**
java.math	**BigDecimal**
java.math	**BigInteger**
java.net	**HttpURLConnection**
java.text	**CollationKey**
java.util	**everything**

変更点の詳細

- **File** クラス、**BigInteger** クラス、**BigDecimal** クラス、**Date** クラス、および **CollationKey** クラスに、**compareTo** メソッドが追加されました。
- **java.lang.Character** に、Unicode サブセットを定義した内部クラスが含まれています。
- **java.io.File** に、新しい関数が 17 個あります。
- **HttpURLConnection** に、新しい関数が 2 個あります。
- **java.util** 内のクラスは、JDK 1.2 レベルに相当する機能を完全にサポートしています。

参照

その他の技術情報

クラス ライブラリのサポート

java.lang.Class クラスのメソッド

Visual J# は、**java.lang.Class** クラスの次のメソッドをサポートしています。

- .NET Framework の Type クラスのインスタンスを、**java.lang.Class** クラスの同等のインスタンスに変換します。

```
public static java.lang.Class Class.FromType(System.Type)
```

- **java.lang.Class** クラスのインスタンスを、.NET Framework の **Type** クラスの同等のインスタンスに変換します。

```
public static System.Type Class.ToType(java.lang.Class)
```

- このメソッドが呼び出された **java.lang.Class** クラスのインスタンスによってカプセル化される、.NET Framework の **Type** クラスのインスタンスを返します。

```
public System.Type Class.ToType()
```

参照

その他の技術情報

クラス ライブラリのサポート

com.ms.vjsharp.text.FormatDefaults クラスのメソッド

Visual J# は、**com.ms.vjsharp.text.FormatDefaults** クラスの次のメソッドをサポートしています。

- 指定された形式スタイルおよびロケールに対して、既定の日付および書式のパターンを設定する場合は、次のメソッドを使用します。**DateFormat.getDateTimeInstance** が呼び出された場合は、この既定のパターンが使用されます。

```
public static final com.ms.vjsharp.text.FormatDefaults.set(Locale locale, int formatSt
yle, String datePattern, String timePattern)
```

- ロケールを設定するための呼び出しに地域が指定されなかったときに、既定の地域を設定する場合は、次のメソッドを使用します。たとえば、`new java.lang.Locale("en", "")` のように、country パラメータに空の文字列を渡して **java.lang.Locale** を作成する場合などが該当します。

```
public static final com.ms.vjsharp.text.FormatDefaults.setDefaultRegion(String langNam
e, String defRegion)
```

langName パラメータは、2 文字の言語コードです。*defRegion* パラメータは、2 文字の国別コードです。

参照

その他の技術情報

クラス ライブラリのサポート

com.ms.wfc.app.Locale クラスのメソッド

次の表は、**com.ms.wfc.app.Locale** クラスにある、使用を推奨されていないメソッドおよび、代わりに使用する新しいメソッドの一覧です。Visual J++ から Visual J# にアップグレードする場合や、このクラスを使用してアプリケーションを新規に記述する場合には、新しいメソッドを使用してください。

使用を推奨されていないメソッド	新しいメソッド
com.ms.wfc.app.Locale.getCountryCode	**com.ms.wfc.app.Locale.getCountryRegionCode**
com.ms.wfc.app.Locale.getCountry	**com.ms.wfc.app.Locale.getCountryRegion**
com.ms.wfc.app.Locale.getAbbrevCountry	**com.ms.wfc.app.Locale.getAbbrevCountryRegion**
com.ms.wfc.app.Locale.getNativeCountry	**com.ms.wfc.app.Locale.getNativeCountryRegion**
com.ms.wfc.app.Locale.getDefaultCountry	**com.ms.wfc.app.Locale.getDefaultCountryRegion**
com.ms.wfc.app.Locale.getEnglishCountry	**com.ms.wfc.app.Locale.getEnglishCountryRegion**
com.ms.wfc.app.Locale.getISO3116CountryName	**com.ms.wfc.app.Locale.getISO3116CountryRegionName**

新しく導入されたメソッドのシグネチャは、対応する使用を推奨されていないメソッドと同じです。また、用意されている機能も同じです。

参照

その他の技術情報

クラス ライブラリのサポート

内部例外

J# 例外クラスには、他の例外の代わりに新しい例外がスローされた場合に元の例外を保存できるコンストラクタが用意されています。

```
java.lang.Throwable(String message, System.Exception innerException);
java.lang.Exception(String message, System.Exception innerException);
java.lang.Error(String message, System.Exception innerException);
java.lang.RuntimeException(String message, System.Exception innerException);
```

パラメータ

message

エラー メッセージ内で使用する、問題の性質を記述するテキストです。

innerException

.NET Framework 例外を Java 例外として再度スローする際に保存した元の .NET Framework 例外です。

解説

.NET Framework では、例外の階層構造のルートは System.Exception です。この例外型では、内部例外の概念がサポートされています。内部例外は、**catch** ブロックで例外を処理し、新しい例外をスローする必要がある場合に役立ちます。スタックの上位にある呼び出し元が新しい例外をキャッチしたとき、その例外がスローされた理由を確認する必要がある場合は、InnerException プロパティを照会することによって元の例外を取得できます。Visual J# では、追加のパラメータ *innerException* を受け取る、例外コンストラクタの追加のオーバーロードが用意されています。このパラメータは元の例外です。内部例外は、**get_InnerException()** を呼び出すことによって取得できます。

使用例

```
// innerexcept1.jsl
// This example shows a class that wraps a FileNotFoundException in its
// own exception class. The original exception can be recovered using
// the get_InnerException method on the System.Exception class.
import System.*;
import java.io.*;
import System.Collections.*;

class ConstructionException extends System.Exception
{
    public ConstructionException(String message, System.Exception innerException)
    {
        super(message, innerException);
    }
}

class DataStructure
{
    ArrayList m_List;
    public DataStructure()
    {
        m_List = new ArrayList();
    }

    public void Add(int c)
    {
            m_List.Add((System.Object)(Int32)c);
    }


    public static DataStructure ConstructFromFile(String filename) throws ConstructionExcep
tion
    {
        DataStructure ds = new DataStructure();
        try
        {
            int c;
            // Could throw FileNotFoundException.
```

```
            FileReader reader = new FileReader(filename);

            // Could throw IOException.
            while ((c = reader.read()) != -1)
            {
                ds.Add(c);
            }

        }
        catch (IOException e)
        {
            Console.WriteLine( e.get_Message() );
            throw new ConstructionException("Error creating the data structure.", e);
        }
        return ds;
    }
}


class CMain
{
    public static void main()
    {
      try
        {
            // Assume myfile.dat doesn't exist.
            DataStructure.ConstructFromFile("myfile.dat");

        }
        catch(System.Exception e)
        {
        Console.WriteLine("Exception occurred: " + e.get_Message());

            System.Exception innerEx = e.get_InnerException();
            if (innerEx != null)
            Console.WriteLine("Cause: " + innerEx.get_Message() );
        }
    }
}
```

この例は、**java.lang.Throwable** をキャッチする **catch** 句で内部例外を抽出する方法を示しています。すべての例外をキャッチできるように、**java.lang.Throwable** は追加の例外レイヤでラップされているため、**get_InnerException** を 2 度呼び出す必要があります。詳細については、「Visual J# の例外の階層構造」を参照してください。

```
// innerexcept2.jsl

import System.*;
import java.io.*;
import System.Collections.*;

class ConstructionException extends System.Exception
{
    public ConstructionException(String message, System.Exception innerException)
    {
        super(message, innerException);
    }
}

class DataStructure
{
    ArrayList m_List;
    public DataStructure()
    {
        m_List = new ArrayList();
    }

    public void Add(int c)
```

```
    {
        m_List.Add((System.Object)(Int32)c);
    }


    public static DataStructure ConstructFromFile(String filename) throws ConstructionExcep
tion
    {
        DataStructure ds = new DataStructure();
        try
        {
            int c;
            // Could throw FileNotFoundException.
            FileReader reader = new FileReader(filename);

            // Could throw IOException.
            while ((c = reader.read()) != -1)
            {
                ds.Add(c);
            }

        }
        catch (IOException e)
        {
            Console.WriteLine( e.get_Message() );
            throw new ConstructionException("Error creating the data structure.", e);
        }
        return ds;
    }
}


class CMain
{
    public static void main()
    {
        try
        {
            // Assume myfile.dat does not exist.
            DataStructure.ConstructFromFile("myfile.dat");

        }
        catch(java.lang.Throwable t)
        {
            // Get the exception out of the Throwable wrapper.
            System.Exception e = ((System.Exception) t).get_InnerException();
        Console.WriteLine("Exception occurred: " + e.get_Message());

        // Get the true inner exception.
        System.Exception innerEx = e.get_InnerException();
            if (innerEx != null)
            Console.WriteLine("Cause: " + innerEx.get_Message() );
        }
    }
}
```

出力例

```
Could not find file 'file-name'.
Exception occurred: Error creating the data structure.
Cause: Could not find file 'file-name'.
```

出力例

```
Could not find file 'file-name'.
Exception occurred: Error creating the data structure.
Cause: Could not find file 'file-name'.
```

Visual J++ 6.0 との互換性の問題

このセクションでは、クラス ライブラリのサポートに関連する Visual J# と Visual J++ 6.0 の互換性情報について説明します。

- java.net パッケージでサポートされているプロトコル
- com.ms.com パッケージ内のサポートされていない型

参照
その他の技術情報
クラス ライブラリのサポート

java.net パッケージでサポートされているプロトコル

Visual J# では、FTP、HTTP、およびファイル プロトコルだけがサポートされています。SMTP、NNTP、および MailTo プロトコルはサポートされていません。

参照

関連項目

Visual J++ 6.0 との互換性の問題

com.ms.com パッケージ内のサポートされていない型

com.ms.com パッケージ内の次のクラスとインターフェイスはサポートされていません。

- **com.ms.com.LicenseMgr**
- **com.ms.com.LICINFO**
- **com.ms.com.StdCOMClassObject**

参照

関連項目

Visual J++ 6.0 との互換性の問題

Supplemental UI Library

Visual J# に含まれている Supplemental UI Library (VJSSupUILib.dll) を使用すると、アプレットなどの各種アプリケーションを開発する際に、Microsoft Windows と同様な外観と操作性を持たせることができます。

Supplemental UI Library for Visual J# (VJSSupUILib.dll) では、以下のようなコンポーネントがサポートされています。

JApplet	JInternalFrame	JProgressBar	JTextArea
JButton	JLabel	JRadioButton	JTextComponent
JCheckBox	JLayeredPane	JRadioButtonMenuItem	JTextField
JCheckBoxMenuItem	JList	JRootPane	JToggleButton
JColorChooser	JMenu	JScrollBar	JToolBar
JComboBox	JMenuBar	JScrollPane	JToolTip
JComponent	JMenuItem	JSeparator	JTree
JDesktopPane	JOptionPane	JSlider	JViewport
JDialog	JPanel	JSplitPane	JWindow
JFileChooser	JPasswordField	JTabbedPane	
JFrame	JPopupMenu	JTable	

解説

Microsoft Supplemental UI Library for Visual J# は、Java 2 の JFC/Swing 仕様書に記述されている機能の大半をサポートしています。この製品は、Microsoft が独自に開発している製品であり、Sun Microsystems, Inc. によって保証および承認された製品ではありません。Microsoft Supplemental UI Library for Visual J# 2005 を対象に開発されたアプリケーションは、Microsoft Visual J# 再領布可能パッケージ バージョン 2.0 上でのみ実行できます。

Windows 同様の外観と操作性に関連するクラスは **com.ms.vjsharp.swing.plaf.windows** パッケージに入っています。

参照

関連項目

サポートされているクラス ライブラリ
サポートされていないクラス ライブラリと機能
[参照の追加] ダイアログ ボックス

ThreadDeath 互換性クラスの警告

J++ コード (MSJVM) で例外やエラーが発生した場合、それらが子スレッド内で発生し処理されない場合にはスレッドが強制終了され、メインスレッドで発生した場合にはプロセスが強制終了されます。J++ で開発したアプリケーションの場合、この動作を複数の異なる方法で使用する場合がありました。.NET Framework 共通言語ランタイム (CLR : Common Language Runtime) では、処理されない例外に関するポリシーに互換性に影響する変更点が加えられ、どのスレッドで発生した例外であっても、処理されない場合はプロセスが強制終了されます。**java.lang.ThreadDeath** は、java.lang.Thread.stop() メソッドによって使用される Throwable から特化したクラスのうち、スレッドが停止したことをアプリケーションに通知するために最もよく使用されるクラスです。

参照

その他の技術情報

クラス ライブラリのサポート

Visual J# で記述されたアプリケーションでのセキュリティのセマンティクス

Visual J# 2005 より前のリリースでは、Visual J# を使用して記述されたすべてのアプリケーションやコンポーネントを実行するには、コードを完全に信頼する必要がありました。Visual J# 2005 では、Visual J# ライブラリ (vjslib.dll) を使用するコードについては、この制限が緩和されています。Visual J# 2005 では vjslib.dll アセンブリに AllowPartiallyTrustedCallersAttribute 属性が設定されるため、この動作の変更が行われました。Visual J# 2005 では、Visual J# アプリケーションを .NET 部分信頼セマンティクスのスコープ内で実行できます。具体的には、次のシナリオを使用できます。

- ネットワーク共有から起動した場合に、Visual J# アプリケーションが正しく実行される。
- Web ページに埋め込まれた Windows フォーム コントロールが Visual J# Class Libraryを使用する可能性がある。
- 部分的に信頼されているコードの実行をホスト アプリケーションがサポートする環境で Visual J# コードが実行される可能性がある。

ただし、**com.ms.wfc** (vjswfc.dll) や **com.ms.wfc.html** (vjswfchtml.dll) など、その他のライブラリを使用するコードを実行するには完全な信頼が必要です。セキュリティ ポリシーで設定された FullTrust 名前付きアクセス許可セットが与えられていないコード グループでこのようなアプリケーションまたはコンポーネントを実行すると、セキュリティ例外がスローされます。これは、Visual J# にアップグレードされた **com.ms.com**、**com.ms.dll**、**com.ms.lang**、**com.ms.wfc**、または **com.ms.wfc.html** を使用する既存の Visual J++ アプリケーションに当てはまります。

したがって、Visual J# を使用するアプリケーションを記述する場合は、次の点を考慮します。

- MyComputer コード グループのローカル コンピュータから実行されたすべてのアプリケーションには、.NET Framework の既定のセキュリティ ポリシーで設定された FullTrust アクセス許可セットが与えられます。したがって、ローカル コンピュータから実行されたアプリケーションでは問題は発生しません。

  アプリケーションは、インターネット コード グループまたはイントラネット コード グループ内のリモートの場所 (ネットワーク共有など) から実行される場合があります。これらのコード グループから実行されたアプリケーションには .NET Framework の既定のセキュリティ ポリシーで設定された FullTrust アクセス許可セットが与えられないため、Visual J++ ライブラリを使用した場合は、セキュリティ例外が発生して失敗します。

  この問題の回避方法としては、完全に信頼されたリモートの場所からアプリケーションまたはコンポーネントを実行することが考えられます。.NET Framework セキュリティ ポリシーによって完全に信頼されていると見なされるためには、次の条件のいずれかが満たされる必要があります。

  - コンポーネントまたはアプリケーションがキー ペアを使って署名され、このキー ペアを使って署名されたすべてのコンポーネントやアプリケーションに FullTrust 名前付きアクセス許可セットが与えられるように、コンピュータの .NET Framework セキュリティ ポリシーが変更されている。
  - コンポーネントまたはアプリケーションが Authenticode 証明書を使って署名され、この Authenticode 証明書を使って署名されたすべてのコンポーネントやアプリケーションに FullTrust 名前付きアクセス許可セットが与えられるように、コンピュータの .NET Framework セキュリティ ポリシーが変更されている。
  - コンポーネントまたはアプリケーションをホストしている Web サイト (URL) からダウンロードされたコントロールに FullTrust 名前付きアクセス許可セットが与えられるように、コンピュータの .NET Framework セキュリティ ポリシーが変更されている。

Visual J# のアップグレード リファレンス

このセクションには、Visual J++ 6.0 のプロジェクトを Visual J# にアップグレードするときに、アップグレード ウィザードで検出される問題についてのヘルプ トピックが含まれています。

このセクションのトピックでは、Visual J++ 6.0 のプロジェクトを Visual J# にアップグレードするときに識別される問題について説明しています。これらのトピックは、アップグレード プロセスで生成されるアップグレード レポートのリンクからアクセスするために用意されています。

参照

関連項目

Visual J# アップグレード ウィザード
Visual J# のアップグレード レポート
Visual J# バイナリ変換ツール (Java 言語のバイトコードから MSIL への変換用)

その他の技術情報

Visual J++ 6.0 からのアップグレード

java.io.Externalizable インターフェイスには互換性に関する問題がある

Visual J++ 6.0 でシリアル化されたオブジェクトは、オブジェクトのフィールドがプリミティブ型または **java.lang.String** である場合にのみ、Visual J# で逆シリアル化できます。JDK レベル 1.1.4 パッケージ内の他のクラスを参照する参照するオブジェクトは、逆シリアル化できません。ただし、Visual J# でシリアル化されたすべてのオブジェクトは、この制限にかかわらず、Visual J# で逆シリアル化できます。

参照

関連項目

java.io.Serializable インターフェイスには互換性に関する問題がある

java.io.Serializable インターフェイスには互換性に関する問題がある

Visual J++ 6.0 でシリアル化されたオブジェクトは、オブジェクトのフィールドがプリミティブ型または **java.lang.String** である場合にのみ、Visual J# で逆シリアル化できます。JDK レベル 1.1.4 パッケージ内の他のクラスを参照する参照するオブジェクトは、逆シリアル化できません。ただし、Visual J# でシリアル化されたすべてのオブジェクトは、この制限にかかわらず、Visual J# で逆シリアル化できます。

参照

関連項目

java.io.Externalizable インターフェイスには互換性に関する問題がある

java.lang.Class のメソッドには互換性に関する問題がある

java.lang.Class の次のメソッドには、Visual J++ 6.0 と Visual J# との間で互換性に関する問題があります。

java.lang.Class.forName(String *className* **)**

このメソッドでは、クラスの検索に CLASSPATH 環境変数は使用されません。対応する型を格納しているアセンブリの検索と読み込みは、共通言語ランタイムのセマンティクスに従って行われます。詳細については、「クラスを動的に読み込むアプリケーションのアップグレード」を参照してください。

java.lang.Class.getDeclaredMethods

このメソッドが返す項目の順序は、Visual J++ 6.0 とは異なります。このメソッドが返す項目の順序に依存するアプリケーションでは、適切な変更が必要になる場合があります。

java.lang.Class.getMethods

このメソッドが返す項目の順序は、Visual J++ 6.0 とは異なります。このメソッドが返す項目の順序に依存するアプリケーションでは、適切な変更が必要になる場合があります。

java.lang.Class.getFields

このメソッドが返す項目の順序は、Visual J++ 6.0 とは異なります。このメソッドが返す項目の順序に依存するアプリケーションでは、適切な変更が必要になる場合があります。

参照

関連項目

java.lang.ClassLoader の互換性に関する問題
クラスを動的に読み込むアプリケーションのアップグレード
Visual J++ 6.0 との互換性の問題

java.lang.ClassLoader の互換性に関する問題

バイトコードを **Class** オブジェクトに変換するための **ClassLoader** は、サポートされていません。Visual J# では、次のメソッドがサポートされなくなっています。

- **ClassLoader.defineClass**
- **ClassLoader.resolveClass**

Visual J# では、これらのメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

詳細については、「カスタム クラス ローダーを使うアプリケーションのアップグレード」を参照してください。

java.lang.Compiler クラスがサポートされていない

Visual J# では、java.lang.Compiler クラスはサポートされていません。

参照

関連項目

Visual J++ 6.0 との互換性の問題

Visual J++ 6.0 のセキュリティのセマンティクスを使うアプリケーションのアップグレード

java.text.CollationElementIterator クラスがサポートされていない

Visual J# では、**java.text.CollationElementIterator** クラスはサポートされていません。

java.text.RuleBasedCollator の一部のメソッドがサポートされていない

このクラスの **RuleBasedCollator(String)** コンストラクタはサポートされていません。

このクラスでは、次のメソッドもサポートされていません。

- **RuleBasedCollator.getRules**
- **RuleBasedCollator.getCollationElementIterator**

java.lang.Runtime の一部のメソッドには互換性の問題がある

次のメソッドには、Visual J++ 6.0 との互換性の問題があります。

- **java.lang.Runtime.freeMemory**

  Visual J# では、使用できる空きメモリをバイト数で返します。それに対して、Visual J++ 6.0 では、使用できる空きメモリをキロバイト数で返します。Visual J# の動作は、Java クラス ライブラリの仕様に準拠しています。

- **java.lang.Runtime.totalMemory**

  Visual J# では、使用できるメモリの合計をバイト数で返します。それに対して、Visual J++ 6.0 では、使用できるメモリの合計をキロバイト数で返します。Visual J# の動作は、Java クラス ライブラリの仕様に準拠しています。

- **java.lang.Runtime.load**

次のメソッドは、Visual J# ではサポートされていません。

- **java.lang.Runtime.traceInstructions**
- **java.lang.Runtime.traceMethodCalls**

Visual J# では、これらのメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

参照
関連項目
Visual J++ 6.0 との互換性の問題

java.lang.SecurityManager クラスには互換性に関する問題がある

Visual J# では、このクラスのメソッドのうち、**CallStack** を処理するメソッドがサポートされていません。サポートされていないメソッドは次のとおりです。

- **classDepth**
- **classLoaderDepth**
- **currentClassLoader**
- **currentLoadedClass**
- **getClassContext**
- **inClass**
- **inClassLoader**

Visual J# では、これらのメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

詳細については、「Visual J++ 6.0 のセキュリティのセマンティクスを使うアプリケーションのアップグレード」を参照してください。

java.lang.Thread.getName メソッドには互換性に関する問題がある

Visual J# では、**java.lang.Thread.getName** メソッドに互換性の問題があります。「Visual J++ 6.0 との互換性の問題」を参照してください。

参照

関連項目

Visual J# のアップグレード リファレンス

java.lang.Thread.dumpStack メソッドがサポートされていない

Visual J# では、このメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

参照

関連項目

Visual J++ 6.0 との互換性の問題

java.lang.Thread.countStackFrames メソッドがサポートされていない

Visual J# では、このメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

参照

関連項目

Visual J++ 6.0 との互換性の問題

java.lang.ThreadGroup.allowThreadSuspension メソッドがサポートされていない

Visual J# では、このメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

参照

関連項目

Visual J++ 6.0 との互換性の問題

java.security パッケージには互換性に関する問題がある

Visual J# に用意されている既定のセキュリティ プロバイダには、Visual J++ 6.0 の既定のプロバイダとの間で互換性に関する問題があります。

既定のセキュリティ プロバイダによって生成されるキーでは、キーのエンコーディングがサポートされていません。したがって、次の **java.security** メソッドは、Visual J# の既定のプロバイダではサポートされていません。

- **java.security.Key.getEncoded**
- **java.security.Key.getFormat**

Visual J# では、これらのメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

- 既定のセキュリティ プロバイダは、他のセキュリティ プロバイダによって生成されたキーをサポートしていません。このため、サードパーティの秘密キーと公開キーを使った署名およびデータの検査はできません。
- 既定のセキュリティ プロバイダは、他のセキュリティ プロバイダによって生成された署名を検査できません。

ただし、Visual J# では、サードパーティのセキュリティ プロバイダを設定して、キーや署名の生成などの操作に使用できます。

詳細については、「サードパーティのセキュリティ プロバイダを使うアプリケーションのアップグレード」を参照してください。

com.ms.ActiveX パッケージがサポートされていない

Visual J# では、ActiveX コントロールのホストになることはサポートされていません。詳細については、「Java 言語での ActiveX コントロールのホスト」を参照してください。

com.ms.asp パッケージがサポートされていない

詳細については、「com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード」を参照してください。

com.ms.awt パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.com.ComSuccessException クラスには互換性に関する問題がある

Visual J++ 6.0 からアップグレードされたプロジェクトでは、S_FALSE などのエラーでない HRESULT は常に S_OK と同等に扱われ、**com.ms.com.ComSuccessException** 型の例外はスローされません。

参照

関連項目

COM を呼び出す Java 言語

com.ms.com.NoAutoMarshaling インターフェイスには互換性に関する問題がある

Visual J# では、JActiveX ラッパーのマーカー インターフェイスである **com.ms.com.NoAutoMarshaling** および **com.ms.com.NoAutoScripting** はサポートされていません。これらのインターフェイスの機能に依存する Visual J++ アプリケーションでは、アップグレード時に適切な変更が必要になる場合があります。

詳細については、「問題と回避策 : COM を呼び出す Java 言語」を参照してください。

参照

関連項目

COM を呼び出す Java 言語

com.ms.debug パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.directX パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.ie パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.iis パッケージがサポートされていない

詳細については、「com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード」を参照してください。

com.ms.io.clientstorage パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.io パッケージがサポートされていない

com.ms.io パッケージはサポートされていません。その代わりに、**java.io** パッケージまたは **System.componentmodel.license** パッケージを使用できます。詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

参照

関連項目

Visual J# のアップグレード リファレンス

com.ms.license パッケージがサポートされていない

com.ms.license パッケージはサポートされていません。その代わりに、**System.ComponentModel.License** パッケージを使用できます。詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

参照

関連項目

Visual J# のアップグレード リファレンス

com.ms.mtx パッケージがサポートされていない

詳細については、「com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード」を参照してください。

com.ms.net.wininet パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.object パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.object.dragdrop パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.security パッケージがサポートされていない

詳細については、「com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード」を参照してください。

com.ms.security.permissions パッケージがサポートされていない

詳細については、「com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード」を参照してください。

com.ms.service パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.util.InputMethod パッケージがサポートされていない

詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.vm パッケージがサポートされていない

詳細については、「com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード」を参照してください。

com.ms.wfc.app.Locale クラスには互換性に関する問題がある

com.ms.wfc.app.Locale クラスの次のメソッドの使用は推奨されなくなりました。

- **com.ms.wfc.app.Locale.getCountryCode**
- **com.ms.wfc.app.Locale.getCountry**
- **com.ms.wfc.app.Locale.getAbbrevCountry**
- **com.ms.wfc.app.Locale.getNativeCountry**
- **com.ms.wfc.app.Locale.getDefaultCountry**
- **com.ms.wfc.app.Locale.getEnglishCountry**
- **com.ms.wfc.app.Locale.getISO3116CountryName**

Visual J++ アプリケーションを Visual J# にアップグレードする場合は、使用が推奨されなくなったメソッドに代わる新しいメソッドを使う必要があります。詳細については、「com.ms.wfc.app.Locale クラスのメソッド」を参照してください。

参照
関連項目
com.ms.wfc.app.Locale クラスのメソッド

ActiveX コントロール プロジェクトがサポートされていない

Visual J# では、WFC コントロールまたは Java Bean を ActiveX コントロールとして公開することはサポートされていません。Visual J++ プロジェクトが ActiveX コントロールの場合は、Visual J# にアップグレードするとプロジェクトが動作しなくなります。

参照

関連項目

Java 言語を呼び出す COM

com.ms.xml パッケージがサポートされていない

次の com.ms.xml.* パッケージは、Visual J# ではサポートされていません。

- **com.ms.xml.dso**
- **com.ms.xml.om**
- **com.ms.xml.parser**
- **com.ms.xml.util**

アプリケーションを変更して、これらのパッケージのクラスを、同等の機能を持つ .NET Framework ライブラリの **System.Xml** 名前空間のクラスに置き換える必要があります。

詳細については、「com.ms.xml パッケージを使うコンポーネントのアップグレード」を参照してください。

com.ms.com.ComLib.ownsCleanup メソッドの互換性に関する問題

Visual J# では、このメソッドは常に true を返します。

com.ms.dll.DllLib.copy メソッドの互換性に関する問題

Visual J# では、com.ms.dll.DllLib.copy メソッドに互換性の問題があります。「Visual J++ 6.0 との互換性の問題」を参照してください。

com.ms.lang.SystemX の IME (Input Method Editor) に関するメソッドがサポートされていない

com.ms.lang.SystemX クラスの次のメソッドはサポートされていません。

- **getDefaultInputManager**
- **getInputManager**
- **setInputManager**
- **setKeyBoardLanguage**

Visual J# では、これらのメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

com.ms.com.Variant クラスの互換性に関する問題

Visual J# では、VT_BYREF 型の VARIANT のマーシャリングはサポートされていません。

参照

関連項目

COM を呼び出す Java 言語

com.ms.com.NoAutoScripting インターフェイスの互換性に関する問題

Visual J# では、JActiveX ラッパーのマーカー インターフェイスである **com.ms.com.NoAutoMarshaling** および **com.ms.com.NoAutoScripting** はサポートされていません。これらのインターフェイスの機能に依存する Visual J++ アプリケーションでは、適切な変更が必要になる場合があります。

詳細については、「問題と回避策 : COM を呼び出す Java 言語」を参照してください。

参照

関連項目

COM を呼び出す Java 言語

com.ms.security.auditing パッケージがサポートされていない

詳細については、「com.ms.* パッケージから同等の .NET Framework クラス ライブラリへのアップグレード」を参照してください。

com.ms.util.SystemVersionManager クラスがサポートされていない

Visual J# では、**com.ms.util.SystemVersionManager** クラスはサポートされていません。

com.ms.util.IncludeExcludeWildcards クラスがサポートされていない

Visual J# では、**com.ms.util.IncludeExcludeWildcards** クラスはサポートされていません。

com.ms.util.SetComparer クラスがサポートされていない

Visual J# では、**com.ms.util.SetComparer** クラスはサポートされていません。

com.ms.util.IntRanges クラスの互換性に関する問題

Visual J# では、**com.ms.util.IntRanges** クラスの次のメソッドはサポートされていません。

- **IntRanges.condense(IIntRangeComparator** *judge* **)**
- **IntRanges.intersect(IntRanges other**, **IIntRangeComparator** *judge***)**
- **IntRanges.removeRange(int** *i*, **IIntRangeComparator** *hook***)**
- **IntRanges.removeRange(int** *s*, **int** *e*, **IIntRangeComparator** *hook***)**
- **IntRanges.removeRanges(int** *s*, **int** *count*, **IIntRangeComparator** *hook***)**
- **IntRanges.removeSingleton(int** *n*, **IIntRangeComparator** *hook***)**
- **IntRanges.sort(IIntRangeComparator** *judge* **)**
- **IntRanges.compare (IntRanges** *intRange1*, **IIntRangeComparator** *judge***)**

com.ms.util.UnsignedIntRanges クラスの互換性に関する問題

Visual J# では、**com.ms.util.UnsignedIntRanges** クラスはサポートされていません。

com.ms.util.WildcardExpression クラスがサポートされていない

Visual J# では、**com.ms.util.WildcardExpression** クラスはサポートされていません。

com.ms.util.FileVersionInformation クラスの互換性に関する問題

Visual J# では、**com.ms.util.FileVersionInformation** クラスに互換性の問題があります。Visual J# のこのクラスは、**FileType** プロパティと **OperatingSystem** プロパティをサポートしていません。「Visual J++ 6.0 との互換性の問題」を参照してください。

com.ms.util.IWildcardExpressionComparator インターフェイスがサポートされていない

Visual J# では、**com.ms.util.IWildcardExpressionComparator** インターフェイスはサポートされていません。

com.ms.util.IIntRangeComparator インターフェイスがサポートされていない

Visual J# では、**com.ms.util.IIntRangeComparator** インターフェイスはサポートされていません。

com.ms.util.Timer クラスの互換性に関する問題

Visual J# では、**com.ms.util.Timer** クラスの次のコンストラクタはサポートされていません。

- **public Timer(TaskManager** *tm*, **long** *p***)**
- **public Timer(TaskManager** *tm*, **long** *p*, **int** *id***)**
- **public Timer(TaskManager** *tm*, **long** *p*, **boolean** *r***)**
- **public Timer(TaskManager** *tm*, **long** *p*, **int** *id***, boolean** *r***)**
- **public Timer(TaskManager** *tm*, **TimeListener** *defaultListener*, **long** *p***)**
- **public Timer(TaskManager** *tm*, **TimeListener** *defaultListener*, **long** *p*, **int** *id***)**
- **public Timer(TaskManager** *tm*, **TimeListener** *defaultListener*, **long** *p*, **boolean** *r***)**
- **public Timer(TaskManager** *tm*, **TimeListener** *defaultListener*, **long** *p*, **int** *id*, **boolean** *r***)**

com.ms.util.ProvideSetComparisonInfo インターフェイスがサポートされていない

Visual J# では、**com.ms.util.ProvideSetComparisonInfo** インターフェイスはサポートされていません。

com.ms.util. HTMLTokenizer.mark メソッドおよび com.ms.util. HTMLTokenizer.reset メソッドがサポートされていない

Visual J# では、これらのメソッドを使おうとすると **com.ms.vjsharp.MethodNotSupportedException** がスローされます。

com.ms.util.Task クラスがサポートされていない

Visual J# では、**com.ms.util.Task** クラスはサポートされていません。詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.util.TaskManager クラスがサポートされていない

Visual J# では、**com.ms.util.TaskManager** クラスはサポートされていません。詳細については、「サポートされていないクラス ライブラリと機能」を参照してください。

com.ms.wfc.html パッケージには互換性に関する問題がある

Visual J++ 6.0 で分離コード HTML プロジェクトとして作成されたプロジェクトには、互換性の問題があります。詳細については、「com.ms.wfc.html パッケージを使うコンポーネントのアップグレード」を参照してください。

リソースの検索には互換性に関する問題がある

アップグレード ウィザードにより、Visual J++ 6.0 プロジェクトでリソースが使用されていることが検出されました。Visual J# では、実行時にリソースを検索する方法が一部変更されています。この変更は、アプリケーション ウィザードのプロジェクトで使用される WFC リソースと、**java.lang.Class.getResource** メソッドを使って検索されるすべてのユーザー指定リソースに影響します。

Visual J# でアプリケーションのリソースが正しく検索されるようにするには、リソースを変換して、.NET Framework を対象としたアプリケーションのリソース ファイルで使用される新しい形式にする必要があります。

プロジェクトのリソース ファイルのうち、拡張子が .resources および .properties のファイルは、ウィザードによって自動的に新しい形式に変換され、拡張子 .resx を付けて保存されます。これらのリソースに対して処理を行う必要はありません。

それ以外の拡張子を持つ Visual J++ 6.0 プロジェクトのリソース ファイルや、プロジェクト フォルダの外部にある CLASSPATH で指定された場所から読み込まれるリソースは、Visual Studio .NET のリソース ファイルの形式に変換する必要があります。

詳細については、「Visual J# アプリケーションでのリソースの使用」を参照してください。

このリリースでサポートされていないサードパーティのパッケージ

次のパッケージは、Visual J++ 6.0 では使用できましたが、Visual J# では使用できません。

- **netscape.javascript**
- **sun.audio**
- **sun.awt.image**
- **sun.beans.editors**
- **sun.beans.infos**
- **sun.io**
- **sun.misc**
- **sun.net.ftp**
- **sun.net**
- **sun.net.nntp**
- **sun.net.smtp**
- **sun.net.www**
- **sun.net.www.protocol.doc**
- **sun.net.www.protocol.file**
- **sun.net.www.protocol.ftp**
- **sun.net.www.protocol.http**
- **sun.security.acl**
- **sun.security.pkcs**
- **sun.security.util**
- **sun.security.x509**
- **sun.tools.jar**

これらのパッケージのクラスを使っているアプリケーションは、.NET Framework クラス ライブラリの同等の機能、または共通言語ランタイムを対象としたサードパーティ ライブラリの同等の機能を使うように、適切に変更する必要があります。

@com.register ディレクティブの互換性に関する問題

Java 言語コンポーネントの COM 呼び出し可能ラッパー (CCW: COM Callable Wrapper) で公開されるインターフェイスのセットには、Visual J++ 6.0 との互換性がありません。Visual J# では、次のインターフェイスは CCW によって公開されません。

- **IProvideClassInfo2** および **IExternalConnection**
- **IPersist**、**IPersistStreamInit**、および **IPersistStorage** (Microsoft Java Virtual Machine では、クラスが **java.io.Serializable** または **java.io.Externalizable** を実装している場合に公開される)
- クラスのマーカー インターフェイスである **com.ms.com.NoAutoMarshaling** および **com.ms.com.NoAutoScripting** は無視されます。これにより、Visual J++ 6.0 に関する実行時の互換性の問題が発生することがあります。
- 現在、COM クライアントの Java 言語/COM クラスによって発生したイベントのシンクはサポートされていません。

参照

関連項目

問題と回避策 : Java 言語を呼び出す COM
COM を呼び出す Java 言語
@com ディレクティブの部分的なサポート

@com.transaction ディレクティブがサポートされていない

Visual J# では、**@com.transaction** ディレクティブは無視されます。

参照

関連項目

Java 言語を呼び出す COM

@com ディレクティブの部分的なサポート

@com.typeinfo ディレクティブがサポートされていない

Visual J# では、**@com.typeinfo** ディレクティブは無視されます。

参照

関連項目

Java 言語を呼び出す COM

@com ディレクティブの部分的なサポート

カスタム マーシャリングが Java 言語/COM 相互運用機能および J/Direct ではサポートされていない

Microsoft Visual J# では、カスタム マーシャラを使って Java 言語型をネイティブ型にマーシャリングする JActiveX ラッパーはサポートされていません。このようなラッパーを Visual J# コンパイラでコンパイルしようとすると、コンパイル エラーが発生します。

詳細については、「カスタム マーシャリングを使う Java 言語/COM コンポーネントのアップグレード」を参照してください。

参照

関連項目

@com ディレクティブの部分的なサポート

WFC フォームで ActiveX コントロールをホストするときにスレッド モデルに関する問題が発生する可能性がある

WFC フォームで ActiveX コントロールをホストする場合は、ActiveX コントロールのホストになるスレッドのアパートメント モデルを、シングルスレッド アパートメント (STA: Single Threaded Apartment) に設定することが必要な場合があります。通常は、クラスのパブリック静的メソッド `void main(String[] args)` に `System.STAThread` 属性を追加することにより、この要件を満たすことができます。

詳細については、「COM コンポーネントのスレッド モデルの処理」を参照してください。

RNI テクノロジがサポートされていない

Visual J# では、RNI (Raw Native Interface) テクノロジはサポートされていません。

参照

関連項目

サポートされていないクラス ライブラリと機能

JNI テクノロジがサポートされていない

Visual J# では、JNI (Java Native Interface) テクノロジはサポートされていません。

参照

関連項目

サポートされていないクラス ライブラリと機能

クラスまたはインターフェイスのタイプ ライブラリが見つからない

アップグレード ウィザードによるプロジェクト ファイルのアップグレード分析中に、プロジェクトで使用されている COM クラスまたはインターフェイスのタイプ ライブラリが見つかりませんでした。ウィザードは、コンピュータのレジストリで COM クラスまたはインターフェイスの GUID を検索して、タイプ ライブラリを探します。レジストリ情報が見つからなかった GUID の詳細については、この問題に対してウィザードから報告された情報を参照してください。

GUID の検索は、インターフェイスが登録されていないために失敗することがあります。この場合は、インターフェイスを実装しているクラスがアップグレードされたプロジェクトで参照されていれば、アップグレード後に問題が解決されることがあります。

ただし、クラスの CLSID が見つからないために GUID の検索に失敗する場合は、アップグレードされたプロジェクトのビルド時にエラーが発生します。この問題を解決するには、アップグレードを実行するコンピュータで、Visual J++ 6.0 プロジェクトを正常にビルドして実行できることを確認してください。

参照

その他の技術情報

Visual J++ 6.0 からのアップグレード

未解決のクラスまたはインターフェイス参照

Visual J++ 6.0 プロジェクトが参照するクラスまたはインターフェイスの一部が、プロジェクト内で定義されていないか、またはアップグレード ウィザードの [参照をプロジェクトに追加する] ページで指定されたアセンブリの参照に含まれていません。参照が見つからなかったクラスや型の詳細については、レポートでこの問題の情報を参照してください。

アップグレード分析では、アップグレード時に指定された Visual J++ 6.0 プロジェクトの参照とアセンブリの参照を調べるだけでなく、Visual J++ 6.0 プロジェクトのクラスパス設定で定義されているディレクトリとアーカイブの場所を調査して、見つからなかった参照の検索も試みます。

この問題は、Visual J++ 6.0 プロジェクトがソリューション内の別のプロジェクトで定義されているクラスに依存している場合、またはシステムの CLASSPATH 環境変数を使って参照される外部クラス ライブラリに依存している場合に報告されることがあります。

この問題を解決するには、未解決のクラスまたはインターフェイスが格納されているアセンブリを、アップグレードされた Visual J# プロジェクトの参照リストに追加します。アセンブリを追加する方法の詳細については、「プロジェクト参照の追加と削除」を参照してください。

別の方法として、未解決のクラスまたはインターフェイスが格納されているアセンブリへの参照をアップグレード時に指定することもできます。アセンブリを指定する方法については、「アップグレード ウィザード : [参照をプロジェクトに追加する]」を参照してください。

外部参照されるクラスやインターフェイスのソース ファイルがなくても、バイナリ ファイルやアーカイブがある場合は、Visual J# バイナリ変換ツール (jbimp.exe) を使うことにより、クラス ファイル (.class) やアーカイブ ファイル (.cab、.jar、または .zip) を MSIL アセンブリに変換できます。

メモ :

参照の一部が見つからなかった場合は、未解決のクラスやインターフェイスを拡張または実装しているクラスやインターフェイスのアップグレード分析が不完全になることがあります。

クラスパスに見つかった未解決のクラスまたはインターフェイス参照

Visual J++ 6.0 プロジェクトから、プロジェクトで定義されたクラスパスに配置されているクラスまたはインターフェイスへの参照が見つかりました。この識別は、未解決のクラスまたはインターフェイスと名前が一致するクラス ファイル (.class) を検索することによって行われます。

この問題の詳細を参照して、見つからなかった参照が検出されたクラスパスの場所 (ディレクトリまたはアーカイブ ファイル) を特定してください。

参照が見つからない問題を解決するには、見つからなかったクラスやインターフェイスの定義を格納するアセンブリを 1 つ以上作成し、そのアセンブリへの参照をアップグレードされたプロジェクトに追加する必要があります。

ソース ファイルがある場合

見つからなかったクラスまたはインターフェイスの参照のソース ファイルがある場合は、Visual J# クラス ライブラリ プロジェクトを使ってソース ファイルからクラス ライブラリをビルドできます。このクラス ライブラリ ファイル (.dll) は、アップグレードされたプロジェクトの参照に追加できます。アップグレード レポートで検出されたクラスパスの場所を使って、このクラス ライブラリに含める必要のあるクラスを特定してください。

ソース ファイルがない場合

ソース ファイルがなくてもバイナリ ファイルまたはアーカイブがある場合は、Visual J# バイナリ変換ツール (jbimp.exe) を使うことにより、クラス ファイル (.class) やアーカイブ ファイル (.cab、.jar、または .zip) を MSIL アセンブリに変換できます。この MSIL アセンブリは、アップグレードされたプロジェクトの参照リストに追加できます。アップグレード レポートで検出されたクラスパスの場所を使って、このアセンブリに含める必要のあるクラスを特定してください。

参照

処理手順

方法 : Visual Studio (C#、J#) のリファレンスを追加および削除する

タイプ ライブラリへの参照をプロジェクトに追加できない

ウィザードによるアップグレード分析中に、プロジェクトが COM タイプ ライブラリを参照していることが検出されました。しかし、この COM タイプ ライブラリ ファイル (.dll または .tlb) への参照を追加できませんでした。追加できなかったタイプ ライブラリは、アップグレード レポートに表示されます。

この問題を修正するには、Visual J# を使ってタイプ ライブラリへの参照を手動で追加します。詳細については、「プロジェクト参照の追加と削除」を参照してください。

ソース管理の設定が新しいプロジェクト用にアップグレードされていない

Visual J++ 6.0 プロジェクトがソース管理の対象になっています。Visual J++ 6.0 プロジェクト ファイルのソース管理設定は、Visual J# プロジェクト ファイルにアップグレードされません。

[ソース管理] メニューの [ソース管理の変更] を使うと、プロジェクトをそのサーバーの場所に再バインドできます。詳細については、「接続の変更」を参照してください。

Visual J# ではクラスパスがサポートされていない

Visual J++ 6.0 プロジェクトに、プロジェクト固有のクラスパス設定が含まれています。このプロパティは、アップグレード分析中にプロジェクトで参照されているクラスを検索するために使用されました。ただし、Visual J# ではクラスパスがサポートされていないため、この設定はアップグレードされません。

参照

関連項目

サポートされていない CLASSPATH 変数

ビルド前およびビルド後の設定がアップグレードされない

Visual J++ 6.0 プロジェクトのビルド前およびビルド後の設定は、新しいプロジェクトにアップグレードされません。Visual J# プロジェクトでは、ビルド前およびビルド後の設定はサポートされていません。

これらの設定を置き換えるには、アップグレードされたプロジェクトにカスタム ビルド ステップを作成します。Visual J# プロジェクトでカスタム ビルド ステップを使う場合は、C++ メイクファイル プロジェクトをソリューションに追加し (「メイクファイル プロジェクトの作成」を参照)、[構成の種類] プロパティを [ユーティリティ] に変更します (「[全般] プロパティ ページ (プロジェクト)」を参照)。詳細については、「カスタム ビルド ステップとビルド イベントについて」を参照してください。

プロジェクトのタイプ ライブラリの設定がアップグレードされない

Visual J++ 6.0 プロジェクトでタイプ ライブラリを作成するための設定は、Visual J# にアップグレードされません。Visual J# プロジェクトでは、プロジェクト出力の一部としてタイプ ライブラリを作成することはサポートされていません。

カスタム起動プログラムによりプロジェクトが実行される

Visual J++ 6.0 プロジェクトに、カスタム起動プログラムの設定が含まれています。Visual J++ 6.0 開発環境からアプリケーションを実行すると、この設定で指定された外部プログラムを使ってアプリケーションが起動されます。

Visual J# では、プロジェクト出力がクラス ファイルではなくマネージ アセンブリになっているため、同じ外部プログラムを使うとアプリケーションを起動できない場合があります。

このプロジェクトを Visual J# 開発環境から起動するには、アップグレードされたプロジェクトの "スタート アプリケーション" プロパティを手動で設定して、適切な外部プログラムを指定する必要があります。詳細については、
「[デバッグ] ([<プロジェクト名> プロパティ ページ] ダイアログ ボックス - [構成プロパティ]) (デバイス)」を参照してください。

リソース ファイルのアップグレードの失敗

Visual J++ 6.0 プロジェクトに、Visual J++ 6.0 形式のリソース ファイル (ファイル拡張子が .resources および .properties) が含まれています。アップグレード ウィザードは .NET リソース ファイル形式 (.resx) への変換を試みましたが、予期しないエラーが発生しました。

この問題を解決するには、リソース ファイルを手動で変換し、アップグレードされたプロジェクトに追加します。詳細については、「Visual J# アプリケーションでのリソースの使用」を参照してください。

プロジェクト ファイルの分析が完了しない

アップグレード ウィザードは、Visual J++ 6.0 プロジェクトを分析してアップグレードの問題を検出します。分析中にエラーが発生したため、アップグレード分析が完了しませんでした。このため、一部のアップグレードの問題が検出されず、アップグレード レポートに追加されていない可能性があります。すべてのアップグレードの問題を検出して修正しないと、アップグレードされたプロジェクトのビルドと実行に失敗することがあります。

分析中にエラーが発生する場合は、次のどちらかが原因として考えられます。

Visual J++ 6.0 のプロジェクトのビルド エラー

Visual J++ 6.0 プロジェクトに構文エラーやその他のエラーが含まれているために、Visual J++ 6.0 でのビルドが失敗する場合は、アップグレード分析中にもエラーが発生します。正しくアップグレードするには、Visual J++ 6.0 でプロジェクトがビルドされて実行されることを確認してから、アップグレードをやり直してください。

アップグレード時に参照が見つからない

プロジェクトが外部のクラスや COM オブジェクトを参照している場合は、参照が見つからないためにアップグレード分析が完了していない可能性があります。Visual J++ 6.0 プロジェクトを再度アップグレードして、アップグレード ウィザードの [参照をプロジェクトに追加する] ページにすべての外部参照を追加することをお勧めします。外部参照を追加する方法の詳細については、
「アップグレード ウィザード : [参照をプロジェクトに追加する]」を参照してください。

参照

処理手順

方法 : Visual J++ 6.0 プロジェクトのアップグレード

プロジェクトを新しい形式にアップグレードするときに見つかった問題

Visual J++ 6.0 プロジェクト ファイルから Visual J# へのアップグレード中に、アップグレード ウィザードで問題が見つかりました。原因として、Visual J++ 6.0 プロジェクト ファイルと Visual J# プロジェクト ファイルでは、格納されるプロジェクトのプロパティが異なることが考えられます。

問題の解決

アップグレード ウィザードが起動されている場合は、[その他のアップグレード情報] をクリックすると、プロジェクトがアップグレードに失敗した理由と、エラーの解決策が表示されます。それ以外の場合は、プロジェクト ディレクトリにある _UpgradeReport.htm ファイルを、ブラウザを使って手動で開きます。

参照

関連項目

Visual J# のアップグレード リファレンス

JDK レベル 1.1.4 のクラス ライブラリにおけるアップグレードの問題

アップグレード ウィザードにより、JDK レベル 1.1.4 のクラス ライブラリの使用に関連する問題がプロジェクトのソースから検出されました。Visual J# のクラス ライブラリでサポートされているクラスとメソッドの一部には、Visual J++ 6.0 との互換性の問題があります。詳細については、アップグレード レポートで、このカテゴリの問題の情報を参照してください。

Visual J# でサポートされているクラス ライブラリについては、「クラス ライブラリのサポート」を参照してください。

参照

関連項目

サポートされていないクラス ライブラリと機能

Visual J++ 6.0 との互換性の問題

Microsoft 拡張機能におけるアップグレードの問題

アップグレード ウィザードにより、WFC (Windows Foundation Class) や com.ms.* 階層内のその他のパッケージなど、Visual J++ 6.0 の Microsoft 拡張機能の使用に関する問題が検出されました。これらの問題は、Visual J# で拡張機能の一部がサポートされていない場合、または互換性の問題がある場合に発生します。詳細については、アップグレード レポートで、このカテゴリの問題の情報を参照してください。

参照

その他の技術情報

クラス ライブラリのサポート

Visual J# で見つからないサードパーティの拡張機能

JDK レベル 1.1.4 パッケージに対するサードパーティの拡張機能が使用されています。これらの拡張機能 (netscape.* パッケージや sun.* パッケージなど) は、Visual J++ 6.0 では使用できましたが、Visual J# ではサポートされていません。詳細については、アップグレード レポートで、このカテゴリの問題の情報を参照してください。

参照

関連項目

サポートされていないクラス ライブラリと機能

アップグレードで見つかった未解決の COM 参照

COM コンポーネントへの参照がプロジェクトに見つかりましたが、その参照を解決できませんでした。詳細については、アップグレード レポートで、このカテゴリの問題の情報を参照してください。

原因の 1 つとして、アップグレードを実行しようとしたシステムでは、参照されている COM コンポーネントを使用できない可能性があります。この問題を解決するには、Visual J# へのアップグレードを実行するシステムで、Visual J++ 6.0 を使ってプロジェクトをビルドおよび実行できることを確認してください。

参照

その他の技術情報

Visual J++ 6.0 からのアップグレード

アップグレードで見つかった未解決の参照

クラスまたはインターフェイスへの参照がプロジェクトのソースに見つかりましたが、いくつかの参照が解決されませんでした。詳細については、アップグレード レポートで、このカテゴリの問題の情報を参照してください。

参照

その他の技術情報

Visual J++ 6.0 からのアップグレード

その他の問題

アップグレード ウィザードで検出された一般的なアップグレードの問題です。詳細については、Visual J# アップグレード レポートで、このカテゴリの問題の情報を参照してください。

参照

その他の技術情報

Visual J++ 6.0 からのアップグレード

J# のキーワード

J# プログラムで使用できる言語キーワードの一覧を以下に示します。

キーワードは、定義済みの予約されている識別子であり、コンパイラに対して特別な意味を持ちます。プログラムでキーワードを識別子として使用するためには、プレフィックスとして @ を付けます。たとえば、**@if** は有効な識別子ですが、**if** はキーワードであるため有効な識別子ではありません。詳細については、「識別子としてのキーワードの使用」を参照してください。

リファレンス

abstract (Visual J#)	final (Visual J#)	public (Visual J#)
boolean (Visual J#)	try-finally (Visual J#)	return (Visual J#)
break (Visual J#)	float (Visual J#)	short (Visual J#)
byte (Visual J#)	for (Visual J#)	static (Visual J#)
try-catch (Visual J#)	if (Visual J#)	strictfp (Visual J#)
char (Visual J#)	implements (Visual J#)	super (Visual J#)
class (Visual J#)	import (Visual J#)	switch (Visual J#)
const (Visual J#)	instanceof (Visual J#)	synchronized (Visual J#)
continue (Visual J#)	int (Visual J#)	this (Visual J#)
default (Visual J#)	interface (Visual J#)	throw (Visual J#)
delegate (Visual J#)	long (Visual J#)	throws (Visual J#)
do (Visual J#)	multicast (Visual J#)	transient (Visual J#)
double (Visual J#)	native (Visual J#)	true (Visual J#)
enum キーワード	new (Visual J#)	ubyte (Visual J#)
extends (Visual J#)	null (Visual J#)	void (Visual J#)
false (Visual J#)	package (Visual J#)	volatile (Visual J#)
protected (Visual J#)	private (Visual J#)	while (Visual J#)

参照

概念

Visual J# Class Library

アクセス修飾子 (Visual J#)

アクセス修飾子は、メンバまたは型のアクセス レベルを指定するために使用されるキーワードです。ここでは、3 つのアクセス修飾子について説明します。

- public (Visual J#)

  アクセスの制限はありません。

- protected (Visual J#)

  アクセスは、コンテナ クラス、またはコンテナ クラスから派生した型に制限されます。

- private (Visual J#)

  アクセスはコンテナ型に制限されます。

参照

関連項目

J# のキーワード

private (Visual J#)

private キーワードは、メンバ アクセス修飾子です。プライベートなアクセスは、許容度が最も低いアクセス レベルです。プライベート メンバには、メンバが宣言されているクラスの本体内でだけアクセス可能です。

private メンバへの参照を、メンバが宣言されているクラスの外側から行った場合は、コンパイル エラーになります。ただし、クラス内で宣言されたプライベート メンバに内部クラスからアクセスすることはできます。

private と他のアクセス修飾子の比較については、「アクセス修飾子 (Visual J#)」を参照してください。

使用例

この例では、`Employee` クラスにパブリック メンバ `Name` と **private** メンバ `Salary` があります。**public** メンバには直接アクセスできますが、プライベート メンバにはパブリックなメソッド `AccessSalary()` を使ってアクセスする必要があります。

```
// private_keyword.jsl

class Employee
{
    public String name = "xx";
    private double salary = 100.00;

    public double AccessSalary()
    {
        return salary;
    }
}

class MainClass
{
    public static void main(String[] args)
    {
        Employee e = new Employee();

        // Accessing the public field:
        String n = e.name;

        // Accessing the private field:
        double s = e.AccessSalary();
    }
}
```

解説

上の例で、次に示すようなステートメントを使用してプライベートなメンバへの直接アクセスを試みるとします。

```
double s = e.salary;
```

次のエラー メッセージが表示されることになります。

```
Cannot access field 'Employee.salary'
```

参照

関連項目

J# のキーワード

その他の技術情報

アクセス修飾子 (Visual J#)

アクセス修飾子 (Visual J#)

protected (Visual J#)

protected キーワードは、メンバ アクセス修飾子です。**protected** メンバには、プロテクト メンバが宣言されているクラス、このメンバが宣言されているクラスの派生クラス、およびクラス パッケージからアクセスできます。

基本クラスの **protected** メンバに派生クラスでアクセス可能なのは、派生したクラス型を使ってアクセスが行われる場合だけです。たとえば、次のコードがあるとします。

```
package P1;
public class A
{
    protected int x = 123;
}

package P2;
import P1.*;
public class B extends A
{
    void F()
    {
        A a = new A();
        B b = new B();
        a.x = 10;    // Error
        b.x = 10;    // OK
    }
}
```

`a.x =10` というステートメントでエラーが発生します。B は A を拡張しているものの、所属するパッケージが異なるためです。

プロテクト メンバを持つクラスのサブクラスであり、かつ、同じパッケージに属しているクラスのみ、プロテクト メンバにアクセスできます。

詳細については、「アクセス修飾子 (Visual J#)」を参照してください。

使用例

次の例では、`MyDerivedC` クラスが `MyClass` の派生クラスです。このため、基本クラスのプロテクト メンバに、派生クラスから直接アクセスできます。

```
// protected_keyword.jsl

class MyClass
{
    protected int x;
    protected int y;
}

class MyDerivedC extends MyClass
{
    public static void main(String[] args)
    {
        MyDerivedC mC = new MyDerivedC();

        // Direct access to protected members:
        mC.x = 10;
        mC.y = 15;
        System.out.println("x = " + mC.x + ", y = " + mC.y);
    }
}
```

出力

```
x = 10, y = 15
```

x および y のアクセス レベルを **private** に変更すると、コンパイラがエラー メッセージを発行します。

```
Cannot access field 'MyClass.x'
Cannot access field 'MyClass.y'
```

参照

関連項目

J# のキーワード

その他の技術情報

アクセス修飾子 (Visual J#)

アクセス修飾子 (Visual J#)

public (Visual J#)

public キーワードは、型および型メンバのためのアクセス修飾子です。パブリックなアクセスは、許容度が最も高いアクセス レベルです。パブリック メンバへのアクセスに制限はありません。

詳細については、「アクセス修飾子 (Visual J#)」を参照してください。

使用例

次の例では、`MyClass1` および `MyClass2` という 2 つのクラスが宣言されています。`MyClass1` のパブリック メンバ x と y は、`MyClass2` から直接アクセスされます。

```
// Access_modifiers_public.jsl
// Public access

class MyClass1
{
    public int x;
    public int y;
}

class MyClass2
{
    public static void main(String[] args)
    {
        MyClass1 mC = new MyClass1();

        // Direct access to public members:
        mC.x = 10;
        mC.y = 15;
        System.out.println("x = " + mC.x + ", y = " + mC.y);
    }
}
```

出力

```
x = 10, y = 15
```

public アクセス レベルを **private** に変更すると、次のエラー メッセージが表示されることになります。

```
Cannot access field 'MyClass1.x'
Cannot access field 'MyClass1.y'
```

参照

関連項目

J# のキーワード

その他の技術情報

アクセス修飾子 (Visual J#)
修飾子のキーワード (Visual J#)
アクセス修飾子 (Visual J#)

例外処理のキーワード (Visual J#)

J# は、プログラムの実行中に発生する例外という異常事態を処理する機構を組み込んでいます。このような例外は、通常の制御の流れの外で処理されます。**try**、**throw**、**throws**、**catch**、**finally** の各キーワードを使用して、例外処理を実装できます。

ここでは、例外処理に関する次のトピックについて説明します。

- throw (Visual J#)
- throws (Visual J#)
- try-catch (Visual J#)
- try-catch-finally (Visual J#)
- try-finally (Visual J#)

参照

関連項目

J# のキーワード

throw (Visual J#)

throw ステートメントは、プログラムの実行中に例外という異常事態が発生したことを通知するために使用します。形式は次のとおりです。

```
throw expression;
```

パラメータ

expression

例外オブジェクト。

解説

スローされる例外は、`NullPointerException` など、**Exception** (**Throwable** のサブクラス) から派生したクラスのオブジェクトです。

次の例のようにして、ユーザー定義の例外をスローさせることもできます。

```
class MyException extends Exception {}
...
throw new MyException();
```

throw ステートメントは、**try-catch** ステートメントまたは **try-finally** ステートメントで使用するのが一般的です。例外がスローされると、プログラムは、この例外を処理する **catch** ステートメントを検索します。

また、**throw** ステートメントを使用して、キャッチした例外を再びスローすることもできます。使用例を含む詳細については、「**throws**」および「**try-catch**」を参照してください。

使用例

throw ステートメントで例外をスローする例を次に示します。

```
// throw example 1

public class ThrowTest
{
    public static void main(String[] args)
    {
        String s = null;

        if (s == null)
        {
            throw new NullPointerException();
        }

        System.out.println("The string s is null");   // not executed
    }
}
```

出力

```
//The NullPointerException exception occurs. You also get
// the following message:
java.lang.NullPointerException
   at ThrowTest.main(String[] args)
```

throw ステートメントと **throws** ステートメントでユーザー定義の例外をスローする例を次に示します。

```
// throw example 2

// Declare the subclass MyException:
class MyException extends Exception
{}
public class ThrowTest1
```

```
{
// Use the throws keyword with the method that throws the exception:
    public static void main(String[] args) throws MyException
    {
        String s = null;

        if (s == null)
        {
            throw new MyException();
        }
    // Not executed.
        System.out.println("The string s is null");
    }
}
```

出力

```
//The MyException exception occurs. You also get the following message:
MyException
   at ThrowTest1.main(String[] args)
```

参照

関連項目

J# のキーワード

throws (Visual J#)

try-catch-finally (Visual J#)

その他の技術情報

例外処理のキーワード (Visual J#)

throws (Visual J#)

throws キーワードは、メソッドでスローされる可能性のある、**Error** や **RuntimeException** から派生する以外の例外を、メソッドの宣言で指定するときに使用します。

解説

プログラムの異常終了を防ぐには、例外がスローされる可能性のあるメソッドの宣言で **throws** を使用するか、**try-catch** ステートメントまたは **try-catch-finally** ステートメントを使って例外を処理します。

特定のメソッドでスローされる可能性のある例外を確認するには、**throws** 句を使わずにプログラムをコンパイルします。このとき、コンパイラによって生成されるエラー メッセージを見れば、宣言またはキャッチする必要のある例外の名前を確認できます。

例 1

次の例では、キーボードから入力された文字を、**System.in.read** メソッドを使用して読み取ります。このメソッドには、**throws** 句が必要です。指定しないと、**java.io.IOException** 例外がスローされます。**throws** 句を指定せずに、この例を実行すると、コンパイル時に "`Exception 'java.io.IOException' is not caught and does not appear in throws clause`" というエラー メッセージが表示されます。

```
// Throws1.jsl
// throws example
import java.io.*;

class MyClass
{
    // Use the throws keyword with the method that is expected to throw
    // an exception:
    public static void main(String[] args) throws IOException
    {
        System.out.print("Please enter a character: ");
        // Read a character from the keyboard:
        char c = (char)System.in.read();
        // Display the character:
        System.out.println("You entered the charcater " + c);
    }
}
```

入力

```
w
```

出力例

```
Please enter a character: w
You entered the character w
```

例 2

次の例では、上と同様のコードで、**try-catch** ステートメント内に **System.in.read** を使用し、プログラムの異常終了を防ぐ方法を示しています。**java.io.IOException** 例外をキャッチするために、**catch** ブロックが使用されています。なお、この場合はスーパー クラス **java.lang.Exception** を使用することもできます。

```
// Throws2.jsl
// throws example
import java.io.*;

class MyClass
{
    public static void main(String[] args)
    {
        // Put the code that may throw an exception inside a try-catch block:
       try
        {
            System.out.print("Please enter a character: ");
```

```
            // Read a character from the keyboard:
            char c = (char)System.in.read();
            // Display the character:
            System.out.println("You entered the character " + c);
        }

        // The catch block may handle the exception or may be empty:
        catch (IOException e)
        {
            System.out.print("Exception caught!");
        }
    }
}
```

入力

e

出力例

```
Please enter a character: e
You entered the character e
```

参照

関連項目

J# のキーワード
throw (Visual J#)
try-catch (Visual J#)
try-catch-finally (Visual J#)

その他の技術情報

例外処理のキーワード (Visual J#)

try-catch (Visual J#)

try-catch ステートメントは、**try** ブロックと、それに続く 1 つ以上の **catch** ブロックで構成されます。**catch** ブロックには、各種の例外を処理するためのハンドラを指定します。このステートメントの形式は、次のいずれかになります。

```
try try-block
catch (exception-declaration-1) catch-block-1
catch (exception-declaration-2) catch-block-2
...
try try-block catch catch-block
```

パラメータ

try-block

例外の発生が予想されるコード セグメントを含みます。

exception-declaration, exception-declaration-1, exception-declaration-2

例外オブジェクト宣言。

catch-block, catch-block-1, catch-block-2

例外ハンドラ。

解説

try-block には、例外を発生させる可能性がある保護されたコード ブロックが含まれます。ブロックは、例外がスローされるか、ブロックが正常に終了するまで実行されます。たとえば、次のコードでは、文字列型の引数でプログラムを実行しようとした場合、NumberFormatException 例外がスローされます。また、引数を指定せずに実行した場合は、ArrayIndexOutOfBoundsException 例外がスローされます。

```
public static void main(String argv[])
{
    try
    {
        int x = Integer.parseInt(argv[0]);
    }
...
}
```

catch 句には、**java.lang.Exception** クラス (またはそのサブクラス) から派生したオブジェクトを引数として指定できます。したがって、次のようにすると、NumberFormatException など、特定の例外を指定してキャッチできます。

```
catch (NumberFormatException e)
{
    // Exception handling statements.
}
```

特定の **catch** 句は、同一の **try-catch** ステートメントで複数使用できます。この場合、**catch** 句は順序どおりにチェックされるため、**catch** 句の順序が重要になります。次の例に示すように、下位の例外、つまり汎用性の高い例外から順にキャッチする必要があります。

```
try
{
    int x = Integer.parseInt(argv[0]);
}
catch (ArrayIndexOutOfBoundsException e)
{
    System.out.println(e);
}
catch (NumberFormatException e)
{
    System.out.println(e);
}
```

```
    // The least specific.
catch (Exception e)
{
    System.out.println(e);
}
```

throw ステートメントは、**catch** ステートメントでキャッチされた例外を再びスローするために **catch** ブロックで使用できます。次に例を示します。

```
catch (Exception e)
{
    // Rethrowing exception e
    throw e;
}
```

try ブロック内では、そこで宣言されている変数だけを初期化します。外部で宣言された変数を初期化すると、ブロックの実行が完了する前に例外が発生する可能性があるためです。たとえば、以下のコード例では、変数 `x` が **try** ブロック内で初期化されます。この変数に **try** ブロックの外部にある `print(x)` ステートメントからアクセスしようとすると、未割り当てのローカル変数が使用されていると見なされ、コンパイラ エラーが発生します。

```
public static void main(String argv[])
{
    int x;
    try
    {
    //    Do not do the following.
        x = 123;
        // ...
    }
    catch (Exception e)
    {
        // ...
    }
    // Error: The local variable 'x' was
    // not initialized before use.
    System.out.print(x);
}
```

例 1

例外を発生させる可能性がある `MyMethod()` メソッドへの呼び出しを含む **try** ブロックの例を次に示します。**catch** 句には、単純にメッセージを例外の名前と共に表示する例外ハンドラが指定されています。**throw** 句が `MyMethod()` 内から呼び出された場合、例外が再度スローされ、システムは **catch** ブロックを検索し、その **catch** ブロックでメッセージを表示します。`MyMethod()` は、例外がスローされる前と、例外が再スローされるときの 2 回アクセスされている点に注目してください。

```
// Rethrowing exceptions:
// try-catch example

class MyClass
{
    public static void main(String[] args)
    {
        MyClass mc = new MyClass();
        try
        {
            String s = null;
            mc.MyMethod(s);
        }

        catch (NullPointerException e)
        {
            System.out.println("Exception caught in the catch block: "+
                               e);
        }
    }
```

```
    public void MyMethod(String s) throws NullPointerException
    {
        if (s == null)
        {
             System.out.println("Hello from MyMethod.");
             throw new NullPointerException(
             "\nRethrowing exception from MyMethod; hello again!");
        }
    }
}
```

出力

```
Hello from MyMethod.
Exception caught in the catch block: java.lang.NullPointerException:
Rethrowing exception from MyMethod; hello again!
```

例 2

2 つの **catch** ステートメントを使用する例を次に示します。最初にある、特定性の最も高い例外がキャッチされます。

```
// Ordering catch clauses
// try-catch example

class MyClass
{
    public static void main(String[] args)
    {
        MyClass x = new MyClass();
        try
        {
            String s = null;
            x.MyMethod(s);
        }

        // Most specific:
        catch (NullPointerException e)
        {
            System.out.println("First exception caught." + e);
        }

        // Least specific:
        catch (Exception e)
        {
            System.out.println("Second exception caught." + e);
        }

    }

    public void MyMethod(String s)
    {
        if (s == null)
            throw new NullPointerException();
    }
}
```

出力

```
First exception caught.java.lang.NullPointerException
//In the preceding example, if you start with the least specific catch clause, you will
get this error message:
Catch unreachable: matching exceptions were caught in earlier catch clause
```

参照

関連項目

J# のキーワード

例外処理ステートメント (C# リファレンス)

try-catch-finally (Visual J#)

通常、**catch** および **finally** は、**try** ブロックのリソースを取得して使用する場合に、対で記述されます。**catch** ブロックで例外的な状況を処理し、**finally** ブロックでリソースを解放します。

```
try try-block catch catch-block(s) finally finally-block
```

パラメータ

try-block

例外の発生が予想されるコード セグメントを含みます。

catch-block(s)

例外ハンドラ。

finally-block

例外ハンドラとクリーンアップ コード。

解説

finally キーワードは、例外ブロック (**try-catch**) の後に実行される一連のコードを定義するときに使用します。**finally** ブロックは省略できます。指定する場合は、**try** ブロックや **catch** ブロックの後ろに記述します。**finally** ブロック内のコードは、**try** ブロックの実行結果に関係なく、必ず一度は実行されることが保証されています。正常に実行された場合、**try** ブロックの最後に到達したプログラムの制御は、**finally** ブロックに移動し、通常はそこで、必要な後処理が実行されます。

return、**continue**、または **break** ステートメントによって **try** ブロックを抜ける場合でも、プログラムの制御が **try** ブロックの外に移動する前に必ず、**finally** ブロックの処理が実行されます。

try ブロックで例外が発生した場合、そのメソッド内で発生した例外を処理するための **catch** ブロックが関連付けられていれば、プログラムの制御はまず、**catch** ブロックに移動し、最後に **finally** ブロックに移動します。例外を処理するための **catch** ブロックがその try ブロックに存在しない場合、プログラムの制御は、まず、**finally** ブロックに移動し、その後、例外が処理されるまで、先に呼び出された一連のメソッドを順にさかのぼってゆきます。

使用例

```
// try-catch-finally
public class EHClass
{
   public static void main(String [] args)
   {
      try
      {
         System.out.println("Executing the try statement.");
         throw new NullPointerException();
      }
      catch(NullPointerException e)
      {
         System.out.println("Caught exception #1:" + e);
      }
      finally
      {
         System.out.println("Executing finally block.");
      }
   }
}
```

出力

```
Executing the try statement.
Caught exception #1:java.lang.NullPointerException
Executing finally block.
```

参照

関連項目

J# のキーワード
throw (Visual J#)
try-catch (Visual J#)
try-finally (C# リファレンス)

その他の技術情報

例外処理のキーワード (Visual J#)

try-finally (Visual J#)

finally ブロックは、**try** ブロックに割り当てられたリソースをクリーンアップするのに便利です。制御は、**try** ブロックがどのように終了したかに関係なく、常に **finally** ブロックに移動します。このステートメントの形式は、次のとおりです。

```
try try-block finally finally-block
```

パラメータ

try-block

例外の発生が予想されるコード セグメントを含みます。

finally-block

例外ハンドラとクリーンアップ コード。

解説

catch が **try** ブロックで発生した例外を処理するのに対して、**finally** はその前にある **try** ブロックがどのように終了したかに関係なくコードのステートメント ブロックを実行することを保証します。

return、**continue**、または **break** ステートメントによって **try** ブロックを抜ける場合でも、プログラムの制御が **try** ブロックの外に移動する前に必ず、**finally** ブロックの処理が実行されます。

使用例

例外を発生させる無効な代入ステートメントの例を次に示します。このプログラムを実行すると例外がスローされますが、**finally** 句は必ず実行されます。

```
// try-finally

public class TryFinally
{
    public static void main(String[] args)
    {
        int[] i = new int[3];

        try
        {
            // Invalid assignment to an array element:
            i[5] = 34;
        }

        finally
        {
            // The following statement is always executed:
            System.out.println("Hello from the finally block!");
        }
    }
}
```

出力

```
//When you run the program, it halts as the IndexOutOfRangeException
// exception occurs. The following message is then displayed:
Unhandled Exception: System.IndexOutOfRangeException: Index was outside the bounds of t
he array.
   at TryFinally.main(String[] args)
Hello from the finally block!
//Notice that although the exception was not caught, the
// statement included in the finally block is still executed:
Hello from the finally block!
```

参照

関連項目

J# のキーワード
throw (Visual J#)
try-catch (Visual J#)
try-catch-finally (Visual J#)

その他の技術情報

例外処理のキーワード (Visual J#)

繰り返しステートメント (Visual J#)

繰り返しステートメントを使用すると、ループを作成できます。繰り返しステートメントは、ループ終了条件に応じて、埋め込みステートメントを複数回繰り返して実行します。このステートメントは、ジャンプ ステートメント (Visual J#)を検出した場合を除いて、順序どおりに実行されます。

繰り返しステートメントでは、次のキーワードを使用します。

- do (Visual J#)
- for (Visual J#)
- while (Visual J#)

参照

関連項目

J# のキーワード

do (Visual J#)

do ステートメントは、指定した式が **false** になるまでステートメントまたはステートメントのブロックを繰り返し実行します。形式は次のとおりです。

```
do statement while (expression);
```

パラメータ

expression

boolean 式、または暗黙的に **boolean** に変換される式。この式は、ループ終了条件をテストします。

statement

埋め込みステートメントまたは実行される複数のステートメント。

解説

while ステートメントとは異なり、**do** ステートメント本体のループは、*expression* の値に関係なく最低 1 回は実行されます。

do ループを使用して、5 未満の正数を出力する例を次に示します。

例 1

```
// do1.jsl
// do example

public class TestDoWhile
{
    public static void main(String[] args)
    {
        int x;
        int y = 0;

        do
        {
            x = y++;
            System.out.println(x);
        }
        while(y < 5);
    }
}
```

出力

```
0
1
2
3
4
```

例 2

条件が **false** であってもループが 1 回実行されることの例を次に示します。

```
// do2.jsl
// do example

class DoTest {
    public static void main(String[] args)
    {
        int n = 10;
        do
        {
            System.out.println("Current value of n is " + n);
```

```
            n++;
        } while (n < 6);
    }
}
```

出力

```
Current value of n is 10
```

参照

関連項目

J# のキーワード

その他の技術情報

繰り返しステートメント (Visual J#)

for (Visual J#)

for ループは、指定した式が **false** になるまでステートメントまたはステートメントのブロックを繰り返し実行します。形式は次のとおりです。

```
for ([initializers]; [expression]; [iterators]) statement
```

パラメータ

initializers

ループ カウンタを初期化する式または代入ステートメントをコンマで区切ったリスト。

expression

暗黙的に **bool** に変換できる式。この式は、ループ終了条件をテストします。

iterators

ループ カウンタをインクリメントまたはデクリメントする式ステートメント。

statement

埋め込みステートメントまたは実行される複数のステートメント。

解説

for ステートメントは、次のように *statement* を繰り返し実行します。

- まず、*initializers* が評価されます。
- 次に、*expression* が **true** の間、*statement* が実行され、*iterators* が評価されます。
- *expression* が **false** になると、制御はループの外部に移動します。

for ステートメントは、*expression* がテストされてからループが実行されるので、0 回以上実行されます。

for ステートメントの式はすべてオプションなので、たとえば次のステートメントでは無限ループを記述できます。

```
for (;;)
{
    ...
}
```

使用例

```
// statements_for1.jsl
// for loop example

public class ForLoopTest
{
    public static void main(String[] args)
    {
        for (int i = 1; i <= 5; i++)
            System.out.println(i);
    }
}
```

出力

```
1
2
3
4
5
```

参照

関連項目

J# のキーワード

その他の技術情報

繰り返しステートメント (Visual J#)

while (Visual J#)

while ステートメントは、指定した式が **false** になるまでステートメントまたはステートメントのブロックを実行します。形式は次のとおりです。

```
while (expression) statement
```

パラメータ

expression

boolean 式、または暗黙的に **boolean** に変換される式。この式は、ループ終了条件をテストします。

statement

埋め込みステートメントまたは実行される複数のステートメント。

解説

while ループは、*expression* が評価されてからループが実行されるので、0 回以上実行されます。

while ループは、**break**、**return**、または **throw** ステートメントがループの外部に制御を移動すると終了できます。ループを終了せずに制御を次の繰り返しに移動させるには、**continue** ステートメントを使用します。

使用例

```
// while1.isl
// while example

class WhileTest
{
    public static void main(String[] args)
    {
        int n = 1;

        while (n < 6)
        {
        System.out.println("Current value of n is " + n);
        n++;
        }
    }
}
```

出力

```
Current value of n is 1
Current value of n is 2
Current value of n is 3
Current value of n is 4
Current value of n is 5
```

参照

関連項目

J# のキーワード

その他の技術情報

繰り返しステートメント (Visual J#)

ジャンプ ステートメント (Visual J#)

分岐はジャンプ ステートメントで実行されます。ジャンプ ステートメントは、プログラムの制御をすぐに移動できます。ジャンプ ステートメントでは、次のキーワードを使用します。

- break (Visual J#)
- continue (Visual J#)
- default (Visual J#)
- return (Visual J#)

参照

関連項目

J# のキーワード

break (Visual J#)

break キーワードは、break ステートメントを定義するときに使用します。ラベルが指定されていない場合、最も内側のブロック (**while**、**do**、**for**、**switch** のいずれか) の外に制御を移します。**break** ステートメントにラベルを指定して実行した場合、そのブロックを抜け、ラベルが宣言されている行の直後に制御が移されます。

break ステートメントでは、**try-catch** ブロックの外側に制御を移すこともできます。ラベルの指定された **break** ステートメントが **try** ブロックまたは **catch** ブロック内に置かれ、その **try-catch** ブロックと対応する **finally** ブロックが定義されている場合、**finally** ブロック内の処理が先に実行されます。**finally** ブロックの処理がすべて実行された後、ラベル宣言の直後に出現する最初のステートメントに再び制御が移ります。このステートメントの形式は、次のいずれかになります。

```
break;
break label;
```

パラメータ

label

コードのブロックを指定するラベル。

解説

ラベルは、識別子の後にコロン (:) を付けて指定します。

```
myLabel: Statement(s)
```

ラベル ブロックには、必ず **break** ステートメントが存在する必要があります。

break ステートメントの存在しないラベル ブロックに、プログラムの制御を移すことはできません。

例 1

次の例には、条件付きステートメントに 1 から 100 までをカウントするカウンタがあります。ただし、**break** ステートメントによってループは 4 回で終了します。

```
// statements_break1.jsl
// break example

class BreakTest
{
    public static void main(String[] args)
    {
        for (int i = 1; i <= 100; i++)
        {
            if (i == 5)
                break;
            System.out.print(i + " ");
        }
    }
}
```

出力

```
1 2 3 4
```

例 2

switch ステートメント内での **break** の使用例を次に示します。

```
// statements_break2.jsl
// break and switch
import java.io.*;

class BreakSwitch
```

```
{
    public static void main(String[] args) throws IOException
    {
        System.out.print("Enter your selection (1, 2, or 3): ");
        char n = (char)System.in.read();

        switch(n)
        {
            case '1':
                System.out.println("Current value is " + 1);
                break;
            case '2':
                System.out.println("Current value is " + 2);
                break;
            case '3':
                System.out.println("Current value is "+ 3);
                break;
            default:
                System.out.println("Sorry, invalid selection.");
        }
    }
}
```

入力

```
1
```

出力例

```
Enter your selection (1, 2, or 3): 1
Current value is 1
//If you entered 4, the output would be:
//Enter your selection (1, 2, or 3): 4
//Sorry, invalid selection.
```

例 3

次の例では、**break** ステートメントを使用してループを抜け、指定されたラベルに制御を移します。J# では、この形式が goto の代わりに使用されます。ラベル ブロックには、必ず **break** ステートメントが存在する必要があります。

```
// statements_break3.jsl
// break to a label example

class BreakLabel
{
    public static void main(String[] args)
    {
        Bye:
        for(int i=1; i<=5; i++)
        {
            System.out.println("i=" +i);
            for(int j=1; j<=5; j++)
            {
                System.out.println("j=" +j);
                for(int k=1; k<=5; k++)
                {
                    if(k == 1) System.out.println("k=" + k +
                            ": Hello from the inner loop.");
                    if(k == 2)
                    {
                        System.out.println("k=" + k +
                            ": Hello again from the inner loop.");
                        // Jump after looping twice.
                        break Bye;
                    }
                }
                System.out.println("Not reached.");
```

```
            }
                System.out.println("Not reached.");
            }
        System.out.println("Bye, I am out of here!");
        }
    }
```

出力

```
i=1
j=1
k=1: Hello from the inner loop.
k=2: Hello again from the inner loop.
Bye, I am out of here!
```

参照

関連項目

J# のキーワード

switch (Visual J#)

continue (Visual J#)

ラベルを指定しなかった場合、**continue** ステートメントは、最も内側のループ (while、do、for のいずれか) の最後に制御を移します。ループ内で行われている処理はそこで終了し、再びループの先頭に制御が戻されます。外側のループの最後のステートメントに制御を移す場合は、continue ステートメントにラベルを指定します。この場合、continue の移動先のラベルは、**while**、**do**、**for** のいずれかのループを指し示していることが必要です。移動先がループ ステートメントに限定されるという点で、ラベル付きの break ステートメントとは異なります。

ラベルの指定された **continue** ステートメントが、**try** ブロックまたは **catch** ブロック内に置かれ、その **try-catch** ブロックと対応する finally ブロックが定義されている場合、**finally** ブロック内の処理が先に実行されます。**finally** ブロックの処理がすべて実行された後、再び、ラベル宣言の直後にあるループの最後のステートメントに制御が移ります。形式は次のとおりです。

```
continue;
continue label;
```

例 1

次の例では、カウンタが初期化され、1 から 10 までがカウントされます。式 `(i < 9)` と **continue** ステートメントを組み合わせることで、**continue** から **for** 本体の終わりまでのステートメントが読み飛ばされます。

```
// statements_continue1.jsl
// continue example

class ContinueTest
{
    public static void main(String[] args)
    {
        for (int i = 1; i <= 10; i++)
        {
            if (i < 9)
                continue;
            System.out.println(i);
        }
    }
}
```

出力

```
9
10
```

例 2

次の例では、入れ子になった 2 つのループを使用して、文字列から部分文字列を検索します。ラベル付きの **continue** を使用して、外側のループにある処理をスキップしています。

```
// statements_continue2.jsl
// continue example

public class ContinueWithLabelTest
{
  public static void main(String[] args)
  {
    String searchStr = "This is a string";
    String subStr    = "is";
    boolean foundIt  = false;

    int maxlen = searchStr.length() - subStr.length();

BLOCK1:
    for (int i = 0; i <= maxlen; i++)
    {
      int n = subStr.length();
      int j = i;
```

```
        int k = 0;
        while (n-- != 0)
        {
          if (searchStr.charAt(j++) != subStr.charAt(k++))
          {
            continue BLOCK1;
          }
        }
        foundIt = true;
        break;
      }
      if (foundIt)
      {
        System.out.println("Found it");
      }
      else
      {
        System.out.println("Didn't find it");
      }
    }
  }
```

出力

```
Found it
```

参照

関連項目

J# のキーワード

その他の技術情報

ジャンプ ステートメント (Visual J#)

default (Visual J#)

default キーワードは、**switch** ステートメントで既定のラベルを指定する場合に使用します。

参照

関連項目

J# のキーワード

switch (Visual J#)

return (Visual J#)

return ステートメントは、メソッドの実行を終了し、呼び出し側のメソッドに制御を戻します。オプションの *expression* の値を返すこともできます。メソッドの型が **void** 型の場合、**return** ステートメントは省略できます。このステートメントの形式は、次のとおりです。

```
return [expression];
```

パラメータ

expression

メソッドの戻り値。*expression* に、**void** 型のメソッドを使用することはできません。

使用例

次の例では、メソッド `A()` が変数 `Area` を **double** 値として返します。

```
// statements_return.jsl
// return example

class ReturnTest
{
   static double CalculateArea(int r)
   {
      double area;
      area = r*r*Math.PI;
      return area;
   }

   public static void main(String[] args)
   {
      int radius = 5;
      System.out.println("The area is "+
                         Math.ceil(CalculateArea(radius)));
   }
}
```

出力

```
The area is 79.0
```

参照

関連項目

J# のキーワード

その他の技術情報

ジャンプ ステートメント (Visual J#)

リテラル キーワード (Visual J#)

J# には、以下のリテラル キーワードがあります。

null (Visual J#)

true (Visual J#)

false (Visual J#)

参照

関連項目

J# のキーワード

false (Visual J#)

ブール値 **false** を表します。

使用例

```
// keyword_false.jsl
class MyClass
{
    public static void main(String[] args)
    {
        boolean a = false;
        System.out.println( a ? "yes" : "no" );
    }
}
```

出力

```
no
```

参照

関連項目

J# のキーワード

boolean (Visual J#)

true (Visual J#)

null (Visual J#)

null キーワードは、null 参照を表すリテラル キーワードです。null 参照はオブジェクトを一切参照しません。**null** は参照型変数の既定値です。**null** リテラルは、常に **null** 型になります。

解説

null 型からプリミティブ型への変換は一切できません。

恒等変換を除き、**null** 型への変換はできません。

使用例

演算の対象となる変数の値が null 以外であることを事前にチェックするコード例を次に示します。

```
import java.util.Date;
class MyClass
{
    public static void main(String[] args)
    {
        MyClass class1 = new MyClass();
        class1.printBirthday();
    }

    public void printBirthday()
    {
        System.out.println("Hello, " + name);
        System.out.print("Your birthday is: ");
        if (birthday != null)
            System.out.println(birthday.toString());
        else
            System.out.println("unknown!");
    }

    private String name = "John Smith";
    private Date birthday = null;
}
```

出力

```
Hello, John Smith
Your birthday is: unknown!
```

参照

関連項目

J# のキーワード

その他の技術情報

リテラル キーワード (Visual J#)

true (Visual J#)

ブール値 true を表します。

使用例

```
// js_keyword_true.jsl
class test
{
    public static void main(String[] args)
    {
        boolean a = true;
        System.out.println( a ? "yes" : "no" );
    }
}
```

出力

```
yes
```

参照

関連項目

J# のキーワード

boolean (Visual J#)

false (Visual J#)

その他の技術情報

リテラル キーワード (Visual J#)

修飾子のキーワード (Visual J#)

修飾子は、型および型メンバの宣言を修飾するために使用されます。ここでは、J# の次の修飾子について説明します。

abstract (Visual J#)

const (Visual J#)

final (Visual J#)

multicast (Visual J#)

native (Visual J#)

static (Visual J#)

volatile (Visual J#)

参照

関連項目

J# のキーワード

abstract (Visual J#)

abstract 修飾子は、クラス、メソッド、プロパティ、およびイベントで使用できます。

クラスの宣言に **abstract** 修飾子を使用した場合、クラスは他のクラスの基本クラスとなることだけを目的とします。

解説

抽象クラスには、以下に示す特徴があります。

- 抽象クラスはインスタンス化できません。
- 抽象クラスは、抽象メソッドおよび抽象アクセサを含む可能性があります。
- 抽象クラスを **final** 修飾子で宣言することはできません。
- 抽象クラスから派生される非抽象クラスは、継承される抽象メソッドおよび抽象アクセサの実際の実装をすべて含んでいる必要があります。

abstract 修飾子をメソッド宣言またはプロパティ宣言に使用すると、メソッドまたはプロパティに実装を含まないことを示します。

抽象メソッドには、以下に示す特徴があります。

- 抽象メソッドの宣言は、抽象クラス内でだけ許可されます。
- 抽象メソッドの宣言では実際の実装は用意されないので、メソッドの本体はなく、メソッドの宣言は単にセミコロンで終わり、シグネチャに続くかっこ ({ }) はありません。次に例を示します。

  ```
  public abstract void myMethod();
  ```

- この実装はオーバーライドするメソッドによって用意され、非抽象クラスのメンバとなります。
- **static** 修飾子を抽象メソッド宣言で使用するのはエラーです。

抽象プロパティは抽象メソッドと同様に動作しますが、宣言の構文および呼び出しの構文に相違があります。

- 静的プロパティで **abstract** 修飾子を使用するのはエラーです。
- 抽象プロパティと同じ構文でプロパティ宣言を追加することにより、継承された抽象プロパティを派生クラス内でオーバーライドできます。

抽象クラスは、すべてのインターフェイス メンバの実装を用意する必要があります。

インターフェイスを実装する抽象クラスは、抽象メソッドにインターフェイス メソッドを割り当てることもあります。次に例を示します。

```
interface I
{
    void m();
}
abstract class C implements I
{
    public abstract void m();
}
```

使用例

この例では、`MyDerivedClass` クラスは抽象クラス `MyBaseClass` の派生クラスです。この抽象クラスには `myMethod()` という抽象メソッドと、`get_X()` および `get_Y()` という 2 つの抽象プロパティがあります。

たとえば、次に示すようなステートメントを使用して抽象クラスのインスタンスの作成を試みるとします。

```
    // Error
MyBaseClass mC1 = new MyBaseClass();
```

次のエラー メッセージが表示されることになります。

```
Cannot create a new object of type 'MyBaseClass' because it is an
'abstract' class
```

```
// abstract_keyword.jsl
// Abstract Classes

// Abstract class:
abstract class MyBaseClass
{
    protected int x = 100;
    protected int y = 150;

    // Abstract method:
    public abstract void myMethod();

    // Abstract property:
    /** @property */
    public abstract int get_X();

    // Abstract property:
    /** @property */
    public abstract int get_Y();
}

class MyDerivedClass extends MyBaseClass
{
    public void myMethod()
    {
        x++;
        y++;
    }

    // Overriding property:
    /** @property */
    public int get_X()
    {
        return x+10;
    }

    // Overriding property:
    /** @property */
    public int get_Y()
    {
        return y+10;
    }

    public static void main(String[] args)
    {
        MyDerivedClass mC = new MyDerivedClass();
        mC.myMethod();
        System.out.println("x = " + mC.get_X() +
            ", y = " + mC.get_Y());
    }
}
```

出力

```
x = 111, y = 161
```

参照

関連項目

J# のキーワード

final (Visual J#)

static (Visual J#)

その他の技術情報

修飾子のキーワード (Visual J#)

const (Visual J#)

const はキーワードとして将来使用するために予約されています。

解説

現在、**const** は実装されていませんが、将来使用するために予約されています。そのため、クラス、インターフェイス、フィールド、メソッド、プロパティ、デリゲート、イベントを含む、すべての言語機能において、**const** を識別子として使用することはできません。

参照

関連項目

J# のキーワード

その他の技術情報

アクセス修飾子 (Visual J#)

final (Visual J#)

final 修飾子を使用して、変数、フィールド、メソッド、およびクラスを修飾できます。**final** として宣言された変数は、一度しか設定できません。final フィールドは、クラスのコンストラクタで一度だけ設定できます。**final** 修飾子で宣言されたメソッドをオーバーライドしたり、隠ぺいしたりすることはできません。**Final** のクラスは継承することも、拡張することもできません。

解説

Final 変数の使用

final 修飾子を使用して変数を宣言することにより、ローカル定数を宣言できます。いったん値を代入すると、終始その値が保持されます。**final** 変数は、宣言の最後に初期化する以外に、初期化を保留することもでき、これを "ブランク **final**" と呼びます。

final 変数にオブジェクト参照が格納された場合、オブジェクトに対する操作によって、その状態が変化することはありますが、変数自体は常に同じオブジェクトを参照し続けます。

```
public void myMethod()
{
   // Need not be compile-time constants:
   final int i = (int)(Math.random() * 20);

   // Not yet initialized:
   final int blankfinal;

   // ...
   // Must be initialized before use:
   blankfinal = 101;
   // ...
}
```

Final フィールドの使用

フィールドを **final** として宣言できます。クラス変数およびインスタンス変数は、静的フィールドであるか非静的フィールドであるかにかかわらず **final** として宣言できます。**static** と共に使用した場合、**final** はフラグ定数として使用されます。

使用例

final フィールドの宣言と使用例を次に示します。

```
// final1.jsl
// Constant fields
public class Value
{
    public int i = 1;
}

public class FinalData
{
    // Can be compile-time constants:
    final int i1 = 9;
    static final int i2 = 99;

    // Typical public constant:
    public static final int i3 = 999;

    // Need not be compile-time constants:
    final int i4 = (int)(Math.random() * 11);
    static final int i5 = (int)(Math.random() * 11);

    Value v1 = new Value();
    final Value v2 = new Value();
    static final Value v3 = new Value();

    // Arrays:
    final int[] a = { 1, 2, 3, 4, 5, 6 };
```

```
    public void print(String id)
    {
        System.out.println(id + ": " + "i4 = " + i4 + ", i5 = " + i5);
    }

    public static void main(String[] args)
    {
        FinalData fd1 = new FinalData();
        // Error: Can't change value!
        // fd1.i1++;
        // OK. Object isn't constant
        fd1.v2.i++;
        // OK. Not final.
        fd1.v1 = new Value();

        for (int i = 0; i < fd1.a.length; i++)
        {
            fd1.a[i]++; // OK. Object isn't constant.
        }

        // Error: Can't change handle!
        // fd1.v2 = new Value();
        // Error: Can't change handle!
        // fd1.v3 = new Value();
        // Error: Can't change handle!
        // fd1.a = new int[3];

        fd1.print("fd1");
        System.out.println("Creating new FinalData");
        FinalData fd2 = new FinalData();
        fd1.print("fd1");
        fd2.print("fd2");
    }
}
```

出力例

```
fd1: i4 = 0, i5 = 7
Creating new FinalData
fd1: i4 = 0, i5 = 7
fd2: i4 = 8, i5 = 7
```

Final のメソッドおよびメソッド パラメータの使用

メソッドを **final** として宣言することにより、サブクラスでそのメソッドがオーバーライドされるのを防ぐことができます。プライベート メソッドおよび、**final** クラスで宣言されたすべてのメソッドは、オーバーライドできないため、暗黙的に **final** になります。ただし、プライベート メソッドに、**final** キーワードを追加して冗長的に宣言してもかまいません。**final** として宣言されたメソッドを **abstract** として宣言することはできません。

```
public class MyClass
{
    public final void MyMethod()
    {
    }
}
```

メソッドのパラメータも **final** として宣言できます。この場合、次の例に示すように、メソッド内で、パラメータ ハンドルが指し示す内容を変更することはできません。

```
public class Gyro
{
    public void spin() { }
}

public class FinalParameters
{
```

```
    void withfinal(final Gyro g)
    {
        // Error: g is final.
        //g = new Gyro();
        g.spin();
    }
    void withoutfinal(Gyro g)
    {
        // OK: g is not final.
        g = new Gyro();
        g.spin();
    }

    public int f1(final int i)
    {
        // Error: Can't change i.
        // return ++i;
        // You can only read from a final primitive.
        return i + 1;    // OK.
    }

    public static void main(String[] args)
    {
        FinalParameters fp = new FinalParameters();
        fp.withoutfinal(null);
        fp.withfinal(null);

        int i = 100;
        int j = fp.f1(i);
    }
}
```

Final クラスの使用

クラスの定義がそれ自体で完結しており、サブクラス化する必要性がまったくない場合は、クラスを **final** として宣言できます。その結果、**final** として宣言されたクラスは、サブクラスを一切持つことができません。同様に、**final** として宣言されたクラスのメソッドをオーバーライドすることもできません。**final** として宣言されたクラスが、他のクラスの宣言の **extends** 句に使用されることはありません。また、抽象クラスの実装はそれ自体で完結することはないので、**final** として宣言されたクラスを、**abstract** として宣言することはできません。

```
public final class MyClass
{
   public void MyMethod() { }
}
```

参照

関連項目

J# のキーワード

その他の技術情報

修飾子のキーワード (Visual J#)

multicast (Visual J#)

multicast キーワードと **delegate** キーワードを組み合わせることにより、指定されたメソッドをカプセル化するための参照型を宣言できます。デリゲートは C++ の関数ポインタとほぼ同じですが、タイプ セーフであり、安全です。

multicast および **delegate** の宣言は、次の形式をとります。

```
[attributes] [modifiers] multicast delegate result-type identifier ([formal-parameters]);
```

パラメータ

指定項目 :

attributes (省略可能)

追加の宣言情報。

modifiers (省略可能)

public、**private**、**protected** の各アクセス修飾子を使用できます。

result-type

結果の型。指定したメソッドに対してデリゲートを関連付ける場合は、結果の型と、メソッドの戻り値の型とを一致させる必要があります。

identifier

デリゲート名。

formal-parameters (省略可能)

パラメータ リスト。

解説

delegate を使用すると、関数をパラメータとして渡すことができます。デリゲートはタイプ セーフなので、**delegate** として渡す関数には、**delegate** 宣言と同じシグネチャが必要です。

multicast delegate は、複数の名前付きパラメータを受け取り、一度に登録できるという点で、通常のデリゲートとは異なります。**delegate** を呼び出すと、登録した順にすべてのメソッドが呼び出されます。

デリゲートは、イベントの基礎になります。

使用例

multicast delegate の宣言と使用について、簡単な例を次に示します。

```
// keyword_multicast_1.jsl

import System.Delegate;

// multicast delegate declaration
/** @delegate */
multicast delegate void MyDelegate(int i);

class Program
{
    public static void main(String[] args)
    {
        Program p = new Program();

        MyDelegate d1 = new MyDelegate(p.DelegateFunctionA);
        MyDelegate d2 = new MyDelegate(p.DelegateFunctionB);
        MyDelegate d = (MyDelegate)Delegate.Combine(d1, d2);
        p.TakesADelegate(d);
    }

    public void TakesADelegate(MyDelegate SomeFunction)
```

```
    {
        SomeFunction.Invoke(21);
    }

    public void DelegateFunctionA(int i)
    {
        System.out.println(
            "Called by multicast delegate with number: " + i + ".");
    }

    public void DelegateFunctionB(int i)
        {
            System.out.println(
                "Once again, that number was: " + i + ".");
    }
}
```

出力

```
Called by multicast delegate with number: 21.
Once again, that number was: 21.
```

参照

関連項目

J# のキーワード
delegate (Visual J#)
private (Visual J#)
protected (Visual J#)
public (Visual J#)

その他の技術情報

アクセス修飾子 (Visual J#)

native (Visual J#)

native は、メソッドがアンマネージ コードで作成されていることを宣言する場合に、メソッドの修飾子として使用します。

メソッドの修飾子

native 修飾子をメソッド宣言で使用すると、メソッドが外部で実装されていることを明示できます。**native** 修飾子は、一般に **dll.import** 属性と共に使用します。詳細については、「DllImportAttribute」を参照してください。

abstract 修飾子および **native** 修飾子を一緒に使用して同一のメンバを修飾するのは、エラーです。**native** 修飾子を使用して、メソッドが J# コードの外部で実装されていることを意味します。一方、**abstract** 修飾子を使用すると、メソッドの実装がクラスには用意されていないことを意味します。

外部メソッドの宣言では実際の実装は用意されないので、メソッドの本体はなく、メソッドの宣言は単にセミコロンで終わり、シグネチャに続くかっこ ({ }) はありません。次に例を示します。

```
public static native int myMethod(int x);
```

メモ :

native キーワードの用法は、C++ の **extern** キーワードよりも制限されています。C++ のキーワードとの比較については、『C++ Language Reference』の「Using extern to Specify Linkage」を参照してください。

例 1

この例では、ユーザーの入力したメッセージがプログラムに受け取られ、メッセージ ボックスに表示されます。このプログラムは、User32.dll ライブラリからインポートされた **MessageBox** メソッドを使用します。

```
class MyClass
{
    /** @dll.import("User32.dll") */
    public static native int MessageBox(
        int h,
        String m,
        String c,
        int type);

    public static void main(String[] args)
    {
        try
        {
            byte[] myString = new byte[100];
            System.out.print("Enter your message: ");
            System.in.read(myString);
            int retVal = MessageBox(0, new String(myString).trim(),
                "My Message Box", 0);
        }
        catch (java.io.IOException ex)
        {
            System.out.println(ex.toString());
        }
   }
}
```

例 2

C で記述された外部の DLL を作成する例を次に示します。例 3 では、それを J# プログラムから呼び出しています。

コード

```
// js_cmdll.c
// compile with: /LD /MD
int __declspec(dllexport) myMethod(int i)
{
```

```
    return i*10;
}
```

例 3

native キーワードを使用して、例 2 で作成した外部の DLL を J# プログラムから呼び出す例を次に示します。

```
// js_cm.jsl
public class MyClass
{
    /** @dll.import("js_cmdll.dll") */
    public static native int myMethod(int x);
    public static void main(String[] args)
    {
        System.out.println("myMethod() returns " +
            myMethod(5) + ".");
    }
}
```

出力

```
myMethod() returns 50.
```

解説

このプロジェクトをビルドするには、以下の手順を実行します。

- Visual C++ のコマンド ラインを使用して、`js_cmdll.c` を DLL にコンパイルします。

  ```
  cl /LD /MD js_cmdll.c
  ```

- コマンド ラインを使用して `CM.jsl` をコンパイルします。

  ```
  vjc js_cm.jsl
  ```

これにより js_cm.exe という実行可能ファイルが生成されます。このプログラムが実行されると、`myMethod` が値 5 を DLL ファイルに渡し、DLL は渡された値に 10 を乗算した値を返します。

参照

関連項目

J# のキーワード

abstract (Visual J#)

その他の技術情報

その他のキーワード (Visual J#)

static (Visual J#)

static 修飾子は、静的メンバの宣言に使用します。静的メンバは、特定のオブジェクトではなく、型自体に属するメンバです。**static** 修飾子は、クラス、フィールド、メソッド、プロパティ、およびイベントと組み合わせて使用できますが、デストラクタやクラス以外の型で使用することはできません。

解説

- 定数宣言や型宣言は、暗黙に静的メンバです。
- クラスのインスタンスには同クラスのすべてのインスタンス フィールドのコピーが別に含まれますが、各静的フィールドのコピーは 1 つだけです。
- 静的メソッドまたは静的プロパティ アクセサへの参照には、**this** は使用できません。
- static として宣言できるのは、内部クラスだけです。
- オブジェクトがメモリに読み込まれたときに一度だけ実行される静的な初期化子を、**static** キーワードを使用して作成することもできます。

> **メモ：**
>
> **static** キーワードの用法は、C++ の場合よりも制限されています。C++ のキーワードとの比較については、『C++ Language Reference』の「Static (C++)」を参照してください。

インスタンスのメンバを例示するために、ある企業の従業員を表すクラスを考えてみます。このクラスには、従業員の数を数えるメソッドと、従業員の数を格納するフィールドがあると仮定します。メソッドおよびフィールドは共に、従業員のどのインスタンスにも属していません。代わりに、それらは企業のクラスに属しています。このため、クラスの静的なメンバとして宣言する必要があります。

使用例

この例では、新しい従業員の名前と ID を読み取り、従業員数のカウンタを 1 インクリメントして、新しい従業員の情報および新しい従業員総数を表示します。簡略化のために、この例では現在の従業員数をキーボード入力から読み取っています。実際のアプリケーションでは、ファイルから情報を読み取るようにしてください。

```
// js_static_keyword.jsl
// Static members
public class Employee
{
    public String id;
    public String name;

    public Employee()
    {
    }

    public Employee(String name, String id)
    {
        this.name = name;
        this.id = id;
    }

    public static int employeeCounter;

    public static int AddEmployee()
    {
        return ++employeeCounter;
    }
}

class MainClass extends Employee
{
    public static void main(String[] args)
    {
        try
```

```
        {
            System.out.print("Enter the employee's name: ");
            byte[] name = new byte[100];
            System.in.read(name);
            System.out.print("Enter the employee's ID: ");
            byte[] id = new byte[100];
            System.in.read(id);
            // Create the employee object:
            Employee e = new Employee(new String(name).trim(),
                                      new String(id).trim());
            System.out.print("Enter the current number of employees: ");
            byte[] n = new byte[100];
            System.in.read(n);
            Employee.employeeCounter = Integer.parseInt(
                new String(n).trim());
            Employee.AddEmployee();
            // Display the new information:
            System.out.println("Name: " + e.name);
            System.out.println("ID:   " + e.id);
            System.out.println("New Number of Employees: " +
                               Employee.employeeCounter);
        }
        catch (java.io.IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

入力

```
Jan Stoklasa
AF643G
15
```

出力例

```
Enter the employee's name: Jan Stoklasa
Enter the employee's ID: AF643G
Enter the current number of employees: 15
Name: Jan Stoklasa
ID:   AF643G
New Number of Employees: 16
```

参照

関連項目

J# のキーワード

this (Visual J#)

その他の技術情報

アクセス修飾子 (Visual J#)

volatile (Visual J#)

volatile キーワードは、オペレーティング システム、ハードウェア、現在実行中のスレッドなどによって、フィールドがプログラム中で変更される場合があることを示します。

```
volatile declaration
```

パラメータ
declaration

フィールドの宣言。

解説

システムは、volatile オブジェクトの値が要求されると、前回の命令で同一オブジェクトの値が要求されていた場合でも、新たに要求された時点の値を必ず読み取ります。また、このオブジェクトの値は代入直後に書き込まれます。

volatile 修飾子は、通常、アクセスをシリアル化する **synchronized** ステートメントが使用されない場合に複数のスレッドによりアクセスされるフィールドに対して使用します。**volatile** 修飾子を使用すると、あるスレッドは別のスレッドによって書き込まれた最新の値を取得するようになります。

volatile として修飾されるフィールドは、次の種類に制限されます。

- 参照型
- 値型
- **byte**、**ubyte**、**short**、**int**、**long**、**char**、**float**、**double**、**boolean**

ローカル変数を **volatile** として宣言することはできません。**volatile** として宣言できるのはフィールドだけです。

使用例

次の例では、public のフィールド変数を **volatile** として宣言する方法を示します。

```
// jsharp_volatile.jsl
class Test
{
   public volatile int i;

   Test(int i)
   {
      this.i = i;
   }

   public int getI()
   {
       return i;
   }

   public static void main(String[] args)
   {
       Test test = new Test(22);
       System.out.println("i is guaranteed to be read here: " +
           test.getI());
   }
}
```

出力

```
i is guaranteed to be read here: 22
```

参照
関連項目
J# のキーワード

boolean (Visual J#)
byte (Visual J#)
char (Visual J#)
double (Visual J#)
float (Visual J#)
int (Visual J#)
long (Visual J#)
short (Visual J#)
synchronized (Visual J#)
ubyte (Visual J#)

その他の技術情報

その他のキーワード (Visual J#)

演算子のキーワード (Visual J#)

ここでは、以下に示す、演算子のキーワードについて説明します。

- new (Visual J#)

  オブジェクトの作成

- instanceof (Visual J#)

  オブジェクトの実行時の型と、特定の型との間に互換性があるかどうかをチェックします。

参照

関連項目

J# のキーワード

instanceof (Visual J#)

オブジェクトの実行時の型と、特定の型との間に互換性があるかどうかをチェックします。**instanceof** 演算子は、次の形式の式で使用します。

```
expression instanceof type
```

パラメータ

expression

参照型の式。

type

型。

解説

instanceof 式は、次の条件がどちらも満たされる場合に true を返します。

- *expression* を **null** にすることはできません。
- *expression* を *type* にキャストすることはできます。つまり、`(type)(expression)` という形式のキャスト式が、例外をスローせずに終了します。

instanceof 演算子では、参照変換、ボックス化変換、またはボックス化解除変換だけが考慮されます。ユーザー定義変換などの他の変換は、**instanceof** 演算では考慮されません。

使用例

```
// js_keyword_instanceof.jsl
// The instanceof operator.
class Class1
{
}

class Class2
{
}

public class IsTest
{
    public static void test(Object o)
    {
        Class1 a;
        Class2 b;

        if (o instanceof Class1)
        {
            System.out.println("o is instance of Class1");
            a = (Class1)o;
            // Do something with "a."
        }
        else if (o instanceof Class2)
        {
            System.out.println("o is instance of Class2");
            b = (Class2)o;
            // Do something with "b."
        }
        else
        {
            System.out.println("o is neither Class1 nor Class2.");
        }
    }

    public static void main(String[] args)
```

```
    {
        Class1 c1 = new Class1();
        Class2 c2 = new Class2();
        test(c1);
        test(c2);
        test("a string");
    }
}
```

出力

```
o is instance of Class1
o is instance of Class2
o is neither Class1 nor Class2.
```

参照

関連項目

J# のキーワード

null (Visual J#)

その他の技術情報

演算子のキーワード (Visual J#)

new (Visual J#)

新しいオブジェクトを作成し、必要なメモリを動的に割り当てます。次に例を示します。

```
MyClass  mc = new MyClass();
```

クラスにコンストラクタが定義されていない場合に、**new** 演算子を使用すると、既定のコンストラクタが呼び出されます。

new 演算子によるメモリ割り当てが失敗した場合は、アプリケーションで例外がスローされます。

使用例

new 演算子を使用してクラス オブジェクトを作成し、初期化する例を次に示します。既定値と代入値が表示されます。

```
// operator_new.jsl
// The new operator.

class MyClass
{
    public String name;
    public int id;

    // Default constructor:
    public MyClass ()
    {
    }

    // Parameterized Constructor:
    public MyClass (int id, String name)
    {
        this.id = id;
        this.name = name;
    }

    public static void main(String[] args)
    {
        // Create objects using the default constructor:
        MyClass Employee1 = new MyClass();

        // Display values:
        System.out.println("Default values:");
        System.out.println("   Class members: " + Employee1.name +
                           ", " + Employee1.id);

        // Create objects using the parameterized constructor:
        MyClass Employee2 = new MyClass(1234, "Craig Combel");

        // Display values:
        System.out.println("Assigned values:");
        System.out.println("   Class members: " + Employee2.name +
                           ", " + Employee2.id);
    }
}
```

出力

```
Default values:
   Class members: null, 0
Assigned values:
   Class members: Craig Combel, 1234
```

この例では、文字列の既定値が **null** です。

参照

関連項目

J# のキーワード
class (Visual J#)

その他のキーワード (Visual J#)

ここでは、次の J# キーワードについて説明します。

- extends (Visual J#)

  他のクラスを拡張 (継承) する場合に、クラス宣言で使用します。

- import (Visual J#)

  パッケージで型の使用を許可します。これにより、そのパッケージ内では型を修飾しないで使用できます。

- package (Visual J#)

  スコープを宣言します。

- strictfp (Visual J#)

  すべての式の結果が、単精度フォーマットおよび倍精度フォーマットで表されるオペランドについての IEEE 754 算術規則の範囲内となるような、厳密な制約を追加します。

- synchronized (Visual J#)

  メソッドまたはステートメント ブロックをクリティカル セクションとしてマークします。

- transient (Visual J#)

  フィールドが、オブジェクトの永続状態に含まれないことを指定します。

- void (Visual J#)

  メソッドに戻り値がないことを指定します。

- enum キーワード

  列挙型を宣言します。

参照

関連項目

J# のキーワード

extends (Visual J#)

他のクラスを拡張 (継承) する場合に、クラス宣言で使用します。継承される側のクラスをスーパー クラス、また、継承する側のクラスをサブクラスといいます。サブクラスの宣言は、次の形式をとります。

```
[attributes] [modifiers] class identifier [extends super-class] { class-body }[;]
```

パラメータ

attributes (省略可能)

追加の宣言情報。

modifiers (省略可能)

使用できる修飾子は、abstract、final、private、protected です。

identifier

クラス名。

super-class (省略可能)

このサブクラスが継承する基本クラス。

class-body

クラス メンバの宣言。

解説

継承する側のクラスは、継承される側のクラスのデータ メンバおよびメソッドにアクセスできます。また、他のメンバを独自に追加することにより、スーパー クラスの機能を拡張することもできます。

J# では単一継承のみ使用できます。つまり、1 つのクラスが継承できるスーパー クラスは 1 つだけです。次にその例を示します。

```
class MyClass extends MySuperClass { }
```

使用例

次の例では、`Employee` クラスが、`Citizen` クラスを拡張しています。`MyClass` では、`Employee` クラスをインスタンス化し、そのインスタンスの情報を `GetInfo()` メソッドで取得して社員情報を表示します。Employee クラスでは、さらに、住民情報を取得するために、スーパー クラスの `GetPersonalInfo()` が呼び出されています。

```
// Keyword-extend.jsl
// Inheritance example

// Super class declaration:
class Citizen
{
    String driversLicenseNo = "555-5555-WA";
    String name = "Sally Abolrous";

      public void GetPersonalInfo()
      {
         System.out.println("Name: " + name);
         System.out.println("ID Card Number: " + driversLicenseNo);
      }
}

// The subclass declaration:
class Employee extends Citizen
{
      String companyName = "Lucerne Publishing";
      String companyID = "ENG-007-USA";

      public void GetInfo()
      {
```

```
            // Calling GetPersonalInfo method from the super calss:
            System.out.println("Citizen's Information:");
            GetPersonalInfo();

            System.out.println("\nJob Information:");
            System.out.println("Company Name: " + companyName);
            System.out.println("Company ID: " + companyID);
        }
    }

class MyClass
{
    public static void main(String[] args)
    {
        Employee E = new Employee();
        E.GetInfo();
    }
}
```

出力

```
Citizen's Information:
Name: Sally Abolrous
ID Card Number: 555-5555-WA

Job Information:
Company Name: Lucerne Publishing
Company ID: ENG-007-USA
```

スーパー クラスで、サブクラスと同じ名前のメソッドを呼び出す場合、**super** というキーワードを使用します。たとえば、`GetInfo` という名前の 2 つのメソッドがある場合は、次のようにして、このメソッドをスーパー クラスで呼び出すことができます。

```
super.GetInfo();
```

詳細については、「super (Visual J#)」を参照してください。

参照

関連項目

J# のキーワード

class (Visual J#)

this (Visual J#)

import (Visual J#)

import ディレクティブは、パッケージ内で型の使用を許可します。そのパッケージ内では型を修飾せずに使用できます。**import** ディレクティブの形式は、次のいずれかになります。

```
import package[[.package]...].class;
import package[[.package]...].*;
```

パラメータ

package

使用対象のクラスが格納されているパッケージ。

class

使用するクラス名。

*

個々のクラスを明示的に指定する代わりに、パッケージに存在するすべてのクラスを対象にする場合に指定します。

解説

import ディレクティブを定義すると、パッケージを逐一指定せずに、パッケージ内の型を使用できます。**import** ディレクティブを指定したからといって、指定したパッケージに含まれるすべてのパッケージにアクセスできるわけではありません。

パッケージには、ユーザー定義のパッケージとシステム定義のパッケージの 2 種類があります。ユーザー定義のパッケージとは、ユーザー (開発者) が自分のコードで定義したパッケージのことです。システム定義のパッケージの一覧については、.NET Framework のドキュメントを参照してください。

使用例

クラスに対して **import** ディレクティブを定義する例を次に示します。この例は、pack1.jsl、pack2.jsl、および pack3.jsl の 3 つのファイルで構成されています。最初の 2 つのファイルをライブラリとしてコンパイルし、次に、これらのライブラリへの参照を使用して、3 番目のファイルをコンパイルします。

```
// pack1.jsl
// compile with: /t:library
package Package1;

public class MyClass1
{
    public String toString()
    {
        return "You are in Package1.MyClass1";
    }
}

// pack2.jsl
// compile with: /t:library
package Package2;

public class MyClass2
{
    public String toString()
    {
        return "You are in Package2.MyClass2";
    }
}

// pack3.jsl
// compile with: /r:pack1.dll,pack2.dll
package Package2;

// Imports all classes in package Package1.
```

```
import Package1.*;
// Imports only class MyClass in package Package2.
import Package2.MyClass2;

class Test
{
    public static void main(String[] args)
    {
        MyClass1 class1 = new MyClass1();
        System.out.println(class1.toString());

        MyClass2 class2 = new MyClass2();
        System.out.println(class2.toString());
    }
}
```

出力

```
You are in Package1.MyClass1
You are in Package2.MyClass2
```

参照

関連項目

J# のキーワード

package (Visual J#)

その他の技術情報

その他のキーワード (Visual J#)

package (Visual J#)

package キーワードは、スコープの宣言に使用します。この **package** のスコープを宣言すると、コードを組織化したり、グローバルに一意な型を作成したりできます。

```
package name[.name1] ...];
type-declarations
```

パラメータ

name, name1

パッケージの名前には、任意の有効な識別子を指定できます。パッケージの名前には、ピリオドを使用できます。

type-declarations

パッケージ内では、以下の型を 1 つ以上宣言できます。

- **class**
- **interface**
- **delegate**

解説

パッケージを明示的に指定しなくても、既定の名前空間が作成されます。作成される無名のパッケージは、グローバル パッケージとも呼ばれ、すべてのファイルに存在します。グローバル パッケージ内にある識別子は、名前付きパッケージで利用できます。

J# では、コンパイル時に **/securescoping** コンパイラ オプションを指定して、パッケージの存在するアセンブリにスコープを限定した場合以外、パッケージには暗黙的にパブリック アクセスが割り当てられます。パッケージ内の要素に割り当てることのできるアクセス修飾子の詳細については、public (Visual J#)、private (Visual J#)、protected (Visual J#) の各アクセス修飾子を参照してください。

C++ や C# の名前空間とは異なり、ソース ファイルごとに 1 つのパッケージ宣言しか指定できません。パッケージを入れ子にする場合は、別々のファイルで宣言する必要があります。

パッケージは、2 つ以上の宣言で定義できます。たとえば、次の例では、`MyCompany.Proj1` パッケージの一部として両方のクラスを定義しています。

```
// file MyClass.jsl
package MyCompany.Proj1;

class MyClass
{
}

// file MyClass1.jsl
package MyCompany.Proj1;

class MyClass1
{
}
```

使用例

別のパッケージで静的なメソッドを呼び出す方法の例を次に示します。この例は、package1.jsl および package2.jsl の 2 つのファイルで構成されています。最初のファイルをライブラリとしてコンパイルし、次に、このライブラリを参照として使用して、2 番目のファイルをコンパイルします。

```
// Package1.jsl
// compile with: /t:library
// A sub-package of SomePackage.

package SomePackage.Nested;

public class NestedPackageClass
```

```
{
    public static void sayHello()
    {
        System.out.println("Hello");
    }
}

// Package2.jsl
// compile with /r:Package1.dll
// Package example.
package SomePackage;

public class MyClass
{
    public static void main(String[] args)
    {
        SomePackage.Nested.NestedPackageClass.sayHello();
    }
}
```

出力

```
Hello
```

参照

関連項目

J# のキーワード
class (Visual J#)
delegate (Visual J#)
import (Visual J#)
interface (Visual J#)
private (Visual J#)
protected (Visual J#)
public (Visual J#)

その他の技術情報

アクセス修飾子 (Visual J#)

strictfp (Visual J#)

strictfp キーワードは、クラス、インターフェイス、メソッドのいずれかに適用できます。**strictfp** キーワードを使ってメソッドを宣言した場合、メソッド内のすべてのコードは、厳密な制約に従って実行されます。クラスまたはインターフェイスに対して **strictfp** キーワードを使用した場合、クラス初期化子や入れ子にされた型を含め、クラス内のすべてのコードが厳密に評価されます。ここでいう厳密な制約とは、単精度および倍精度で表現される演算において、すべての式の結果が、IEEE 754 の算術規則によって導かれることを意味します。

解説

J# コンパイラは **strictfp** を有効なキーワードとして認識しますが、実際の計算結果の厳密性は逆に低くなります。パフォーマンス上の理由により、既定では、浮動小数点の算術規則に、共通言語ランタイム (CLR) の実装が使用されます。

例 1

strictfp 修飾子を使用して宣言されたクラスの例を次に示します。

```
// keyword_strictfp.jsl
// Example of precision control with strictfp

public strictfp class MyClass
{
    public MyClass()
    {
    }

    public static void main(String[] args)
    {
        float aFloat = 0.6710339f;
        double aDouble = 0.04150553411984792d;
        double sum = aFloat + aDouble;
        float quotient = (float)(aFloat / aDouble);

        System.out.println("float:  " + aFloat);
        System.out.println("double: " + aDouble);
        System.out.println("sum:    " + sum);
        System.out.println("quotient:    " + quotient);
    }
}
```

出力例

```
float:  0.6710339
double: 0.04150553411984792
sum:    0.71253945297742238
quotient:    16.1673355
```

例 2

strictfp 修飾子を使用して宣言されたメソッドの例を次に示します。

```
// keyword_strictfp_2.jsl
// Example of precision control with strictfp:
public class MyClass2
{
    public float aFloat;
    public double aDouble;

    public MyClass2()
    {
    }

    public strictfp double add(float a, double b)
    {
        return (a + b);
```

```
    }

    public static void main(String[] args)
    {
        MyClass2 myClass2 = new MyClass2();
        myClass2.aFloat = 0.6710339f;
        myClass2.aDouble = 0.04150553411984792d;
        double sum = myClass2.add(myClass2.aFloat, myClass2.aDouble);

        System.out.println("float:  " + myClass2.aFloat);
        System.out.println("double: " + myClass2.aDouble);
        System.out.println("sum:    " + sum);
    }
}
```

出力例

```
float:  0.6710339
double: 0.04150553411984792
sum:    0.71253945297742238
```

参照

関連項目

J# のキーワード

double (Visual J#)

float (Visual J#)

その他の技術情報

その他のキーワード (Visual J#)

synchronized (Visual J#)

synchronized キーワードは、次の 2 とおりの方法で使用できます。

- ステートメント ブロックをクリティカル セクションとしてマークする。指定のオブジェクトに対する相互排他ロックを取得してステートメントを実行し、その後、ロックを解放します。
- メソッドをクリティカル セクションとしてマークする。指定のオブジェクトに対する相互排他ロックを取得してメソッドの処理を実行し、その後、ロックを解放します。

このステートメントの形式は、次のいずれかになります。

```
synchronized(expression) statement_block
access_modifier synchronized return_value method_name(method_params)
```

パラメータ

expression

同期させるオブジェクト。*expression* は参照型である必要があります。

通常、*expression* は、インスタンス変数を保護する場合は **this**、**static** 変数を保護する場合、または指定されたクラスの静的メソッドでクリティカル セクションが発生する場合はクラス オブジェクト (**getClass**) です。

statement_block

クリティカル セクションのステートメント。

access_modifier

ステートメントのアクセス修飾子。

return_value

メソッドの戻り値。

method_name

synchronized キーワードを適用するメソッドの名前。

method_params (省略可能)

メソッドのパラメータ (存在する場合)。

解説

synchronized によって、あるスレッドがコードのクリティカル セクションになっているときは、別のスレッドはクリティカル セクションにはなりません。同期されたコードを別のスレッドが使おうとすると、オブジェクトが解放されるまで待機します。

synchronized キーワードは、ブロックの最初と、ブロックの最後 (Exit) に、Enter を呼び出します。

例 1

J# でのスレッドの簡単な使用例を次に示します。

```
// statements_lock.jsl
import java.lang.Thread;

class ThreadTest implements Runnable
{
    public void run()
    {
        System.out.println("run called");
    }
}

public class MainClass
{
    public static void main(String[] args)
```

```
    {
        try
        {
            ThreadTest b = new ThreadTest();
            Thread t = new Thread(b);
            t.start();
            // Wait for thread to finish.
            t.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

出力

```
run called
```

例 2

次の例では、スレッドおよび **synchronized** が使用されています。**synchronized** ステートメントが存在する限り、ステートメント ブロックはクリティカル セクションであり、`balance` は負の数にはなりません。

```
// statements_synchronized2.jsl
import java.lang.Thread;
import java.util.Random;

class Account implements Runnable
{
    int balance;

    Random r = new Random();

    public Account(int initial)
    {
        balance = initial;
    }

    public void run()
    {
        doTransactions();
    }

    int withdraw(int amount) throws Exception
    {
        // This condition will never be true unless the synchronized
        // statement is commented out:
        if (balance < 0)
        {
            throw new Exception("Negative Balance");
        }

        // Comment out the next line to see the effect of leaving out
        // the synchronized keyword:
        synchronized (this)
        {
            if (balance >= amount)
            {
                System.out.println("Balance before Withdrawal :  " + balance);
                System.out.println("Amount to Withdraw        : -" + amount);
                balance = balance - amount;
                System.out.println("Balance after Withdrawal  :  " + balance);
                return amount;
            }
            else
            {
```

```
// Transaction rejected:
                return 0;

            }
        }
    }

    public void doTransactions()
    {
        try
        {
            for (int i = 0; i < 100; i++)
            {
                withdraw(r.nextInt(100));
            }
        }
        catch (Exception ex)
        {
            System.out.println(ex.toString());
        }
    }
}

class Test
{
    public static void main(String[] args)
    {
        Thread[] threads = new Thread[10];
        Account acc = new Account(1000);
        for (int i = 0; i < 10; i++)
        {
            Thread t = new Thread(acc);
            threads[i] = t;
        }
        for (int i = 0; i < 10; i++)
        {
            threads[i].start();
        }
    }
}
```

例 3

次の例では、**synchronized** ステートメントは使わずに、前の例の **withdraw** メソッドで **synchronized** メソッドのパラメータを使用しています。

```
// statements_synchronized3.jsl
import java.lang.Thread;
import java.util.Random;

class Account implements Runnable
{
    int balance;

    Random r = new Random();

    public Account(int initial)
    {
        balance = initial;
    }

    public void run()
    {
        doTransactions();
    }

    synchronized int withdraw(int amount) throws Exception
```

```
    {
        // This condition will never be true unless the synchronized
        // method modifier is commented out:
        if (balance < 0)
        {
            throw new Exception("Negative Balance");
        }

        if (balance >= amount)
        {
            System.out.println("Balance before Withdrawal :  " + balance);
            System.out.println("Amount to Withdraw        : -" + amount);
            balance = balance - amount;
            System.out.println("Balance after Withdrawal  :  " + balance);
            return amount;
        }
        else
        {
// Transaction rejected:
            return 0;

        }
    }

    public void doTransactions()
    {
        try
        {
            for (int i = 0; i < 100; i++)
            {
                withdraw(r.nextInt(100));
            }
        }
        catch (Exception ex)
        {
            System.out.println(ex.toString());
        }
    }
}

class Test
{
    public static void main(String[] args)
    {
        Thread[] threads = new Thread[10];
        Account acc = new Account(1000);
        for (int i = 0; i < 10; i++)
        {
            Thread t = new Thread(acc);
            threads[i] = t;
        }
        for (int i = 0; i < 10; i++)
        {
            threads[i].start();
        }
    }
}
```

参照

関連項目

J# のキーワード

static (Visual J#)

this (Visual J#)

その他の技術情報

その他のキーワード (Visual J#)

transient (Visual J#)

transient キーワードは、そのフィールドが、オブジェクトの永続状態に属さないことを指定する場合に使用します。**transient** としてマークされたフィールドは、シリアル化時に状態が保存されないため、逆シリアル化時に復元されません。

解説

既定では、クラスの public、protected、private フィールドはすべて、シリアル化処理時に状態が維持されます。**transient** キーワードでマークされたフィールドは、シリアル化から除外されます。

使用例

6 つの整数が使用されたクラスをシリアル化する例を次に示します。3 つの整数はファイルにシリアル化され、**transient** キーワードでマークされた残りの 3 つはシリアル化されません。このファイルに格納されたデータで新しいオブジェクトを作成すると、**transient** 以外のフィールドだけが元の値に復元されます。シリアル化されない属性は、既定値 (整数の場合は 0) で初期化されます。

```
// keyword_transient.jsl
import java.io.*;

public class MyClass implements Serializable
{
    public static void main(String[] args)
    {
        MyClass myClass = new MyClass();

        // Add 100 to all the fields.
        myClass.publicInt = 100;
        myClass.publicTransientInt = 101;

        myClass.set_ProtectedInt(102);
        myClass.set_ProtectedTransientInt(103);

        myClass.set_PrivateInt(104);
        myClass.set_PrivateTransientInt(105);

        // Print out myClass.
        System.out.println("Before serialization...");
        System.out.println("publicInt: " +
            myClass.publicInt);
        System.out.println("publicTransientInt: " +
            myClass.publicTransientInt);
        System.out.println("protectedInt: " +
            myClass.get_ProtectedInt());
        System.out.println("protectedTransientInt: " +
            myClass.get_ProtectedTransientInt());
        System.out.println("privateInt: " +
            myClass.get_PrivateInt());
        System.out.println("privateTransientInt: " +
            myClass.get_PrivateTransientInt());
        System.out.println();

        try
        {
            // Serialize MyClass to a file.
            FileOutputStream fileOutStr =
                new FileOutputStream("C:\\SerializedClass.txt");

            ObjectOutputStream objOutStr =
                new ObjectOutputStream(fileOutStr);
            objOutStr.writeObject(myClass);

            // Create a new instance of MyClass by reading in
            // the serialized myClass from the file created previously.
            FileInputStream fileInStr =
                new FileInputStream("C:\\SerializedClass.txt");
```

```
            MyClass myClass2 = null;

            ObjectInputStream objInStr =
                new ObjectInputStream(fileInStr);
            myClass2 = (MyClass)objInStr.readObject();

            // Print out myClass2.
            System.out.println("After serialization...");
            System.out.println("publicInt: " +
                myClass2.publicInt);
            System.out.println("publicTransientInt: " +
                myClass2.publicTransientInt);
            System.out.println("protectedInt: " +
                myClass2.get_ProtectedInt());
            System.out.println("protectedTransientInt: " +
                myClass2.get_ProtectedTransientInt());
            System.out.println("privateInt: " +
                myClass2.get_PrivateInt());
            System.out.println("privateTransientInt: " +
                myClass2.get_PrivateTransientInt());
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
        catch (ClassNotFoundException ex)
        {
            System.out.println(ex.toString());
        }
    }

    public MyClass()
    {
    }

    /** @property */
    public int get_ProtectedInt()
    {
        return protectedInt;
    }

    /** @property */
    public void set_ProtectedInt(int value)
    {
        protectedInt = value;
    }

    /** @property */
    public int get_ProtectedTransientInt()
    {
        return protectedTransientInt;
    }

    /** @property */
    public void set_ProtectedTransientInt(int value)
    {
        protectedTransientInt = value;
    }

    /** @property */
    public int get_PrivateInt()
    {
        return privateInt;
    }

    /** @property */
    public void set_PrivateInt(int value)
    {
```

```
            privateInt = value;
        }

        /** @property */
        public int get_PrivateTransientInt()
        {
            return privateTransientInt;
        }

        /** @property */
        public void set_PrivateTransientInt(int value)
        {
            privateTransientInt = value;
        }

        public int publicInt = 0;
        public transient int publicTransientInt = 1;

        protected int protectedInt = 2;
        protected transient int protectedTransientInt = 3;

        private int privateInt = 4;
        private transient int privateTransientInt = 5;
    }
```

出力

```
Before serialization...
publicInt: 100
publicTransientInt: 101
protectedInt: 102
protectedTransientInt: 103
privateInt: 104
privateTransientInt: 105

After serialization...
publicInt: 100
publicTransientInt: 0
protectedInt: 102
protectedTransientInt: 0
privateInt: 104
privateTransientInt: 0
```

参照

関連項目

J# のキーワード

その他の技術情報

その他のキーワード (Visual J#)

void (Visual J#)

メソッドの戻り値の型として **void** を使用する場合は、メソッドが値を戻さないことを示します。

void キーワードは、メソッドのパラメータ リストでは指定できません。パラメータや戻り値を持たないメソッドの宣言は、次のとおりです。

```
void MyMethod();
```

参照

関連項目

J# のキーワード

enum キーワード

Microsoft Visual J# 2005 では、開発者が **enum** キーワードを使用して列挙型を作成できます。一組の名前付き定数 (列挙子リスト) で構成されるユーザー定義の列挙型を宣言するために、このキーワードを使用します。列挙型の基になる型は必ず **int** 型です。既定では、最初の列挙子の値は 0 で、後続の列挙子の値は 1 ずつ増加していきます。次に例を示します。

```
enum Day {Sat, Sun, Mon, Tue, Wed, Thu, Fri}
```

この列挙では、`Sat` は `0`、`Sun` は `1`、`Mon` は `2` のように 1 ずつ増加していきます。次のようなコードで、列挙の要素を直接表示できます。

```
System.out.println(Day.Mon);   // Displays Mon.
```

次のようなコードで、名前付き定数に関連付けられている整数値を表示することもできます。

```
System.out.println((int)Day.Mon);   // Displays 2.
```

要素を初期化すると、要素の値を変更できます。たとえば、次のようにします。

```
enum Day {Sat(1), Sun, Mon, Tue, Wed, Thu, Fri}
```

この列挙では、要素の並びは `0` ではなく、`1` から開始します。

複数の要素を初期化することもできます。たとえば、次のようにします。

```
enum DayOff {Sat(7), Sun(1)}
```

この場合、Sat には 7 が関連付けられ、Sun には 1 が関連付けられます。

列挙値はメソッドのパラメータとして渡すことができます。たとえば、次のようなメソッドを定義できます。

```
public static void MyMethod(DayOff d)
{
    // Display the enum element as a string:
    System.out.println(d);
}
```

このメソッドは次のように呼び出します。

```
MyMethod(DayOff.Sat);   // Displays Sat.
```

解説

列挙型は、タイプ セーフです。つまり、**enum** 型と **int** 型との自動変換は行われません。たとえば、次のコードはコンパイル時のエラーになります。

```
DayOff d1 = 1;   // Error.
```

ただし、**enum** 要素をキャストして整数型の変数を作成することはできます。

```
int d1 = (int)DayOff.Sun;   // Ok.
```

enum の派生元の型を知るためには、次のようなステートメントを実行します。

```
System.out.println("The " +
 DayOff.Sat.getClass().ToType() +
 " type is derived from " +
```

```
 DayOff.Sat.getClass().ToType().get_BaseType() +".");
```

上記のステートメントを実行すると、次のように出力されます。

```
The DayOff type is derived from System.Enum.
```

また、次のようなステートメントを実行すると、基底の型を取得することもできます。

```
System.out.println("Underlying type: " + System.Enum.GetUnderlyingType(DayOff.class.ToType(
)));
```

上記のステートメントを実行すると、次のように出力されます。

```
The underlying type is System.Int32.
```

列挙に FlagsAttribute を適用すると、その列挙をビット フィールド、つまり一組のフラグとして扱う必要があることを示すことができます (例 2 を参照してください)。

例 1

enum 型を **switch** ステートメントで使用することもできます。使用例を次に示します。

```
// enum-ex1.jsl
// Using enum with switch.

// Declare an enum type for working days:
enum WorkingDay {Mon(1), Tue, Wed, Thu, Fri};

public class MyClass
{
    public static void main(String [] args)
    {
        // Declare an enum element of the type WorkingDay:
        WorkingDay d = WorkingDay.Fri;
        int dayNum = (int)d;

        // Test the selected day:
        switch (d)
        {
            case WorkingDay.Mon:
                System.out.println("The selected day is " + d + ".");
                System.out.println("The number of the day is " +
                (int)d + ".");
                break;
            case WorkingDay.Tue:
                System.out.println("The selected day is " + d + ".");
                System.out.println("The number of the day is " +
                (int)d + ".");
                break;
            case WorkingDay.Wed:
                System.out.println("The selected day is " + d + ".");
                System.out.println("The number of the day is " +
                (int)d + ".");
                break;
            case WorkingDay.Thu:
                System.out.println("The selected day is " + d + ".");
                System.out.println("The number of the day is " +
                (int)d + ".");
                break;
            case WorkingDay.Fri:
                System.out.println("The selected day is " + d + ".");
                System.out.println("The number of the day is " +
                dayNum + ".");
                break;
            default:
```

```
                System.out.println("Not case is suitable!");
                break;
            }
        }
    }
```

出力

```
The selected day is Fri.
The number of the day is 5.
```

例 2

この例では、列挙体で **FlagsAttribute** 属性を使用する方法を示します。

```
// enum-ex2.jsl
// Using FlagsAttribute with enum.

import System.*;

/** @attribute Flags() */
public enum CarOptions
{
    SunRoof(0x01),
    Spoiler(0x02),
    FogLights(0x04),
    TintedWindows(0x08)
}

public class flagtest
{
    public static void main(String args[])
    {
        CarOptions options = CarOptions.SunRoof | CarOptions.FogLights;
        System.out.println(options);
        System.out.println((int)options);
    }
}
```

出力

```
SunRoof, FogLights
5
```

FlagsAttribute 属性を削除した場合、上記例の出力は次のようになります。

```
5
5
```

参照

関連項目

J# のキーワード

プリミティブ型 (Visual J#)

ここでは、J# でプリミティブ型を宣言するときに使用するキーワードについて説明します。

- boolean (Visual J#)
- byte (Visual J#)
- char (Visual J#)
- double (Visual J#)
- float (Visual J#)
- int (Visual J#)
- long (Visual J#)
- short (Visual J#)
- ubyte (Visual J#)

参照

関連項目

J# のキーワード

boolean (Visual J#)

boolean キーワードは、真理値を保持するプリミティブ データ型を宣言するときに使用します。**boolean** 型には、リテラルの **true** および **false** で表される 2 つの値があります。

リテラル

boolean 変数にはブール値を代入できます。**boolean** として評価される式を **boolean** 変数に代入することもできます。

```
// keyword_boolean.jsl
package MyPackage;
public class MyClass
{
    public static void main(String[] args)
    {
        boolean i = true;
        char c = '0';
        System.out.println(i);
        i = false;
        System.out.println(i);

        boolean alphabetic = (c > 64 && c < 123);
        System.out.println(alphabetic);
    }
}
```

出力

```
true
false
false
```

変換

C++ では、**boolean** 型の値を **int** 型の値に変換できます。つまり、**false** は 0 と等価であり、**true** は 0 以外の値と等価です。J# では、**boolean** 型とその他の型は変換できません。たとえば、次の **if** ステートメントは、C++ では有効ですが、J# では無効です。

```
int x = 123;
    // Invalid in J#
if (x)
{
    printf_s("The value of x is nonzero.");
}
```

int 型の変数をテストするには、次のように、明示的に特定の値 (たとえば 0) と比較する必要があります。

```
int x = 123;
// The J# way:
if (x != 0)
{
    System.out.println("The value of x is nonzero.");
}
```

例 1

ここでは、キーボードから文字が入力されると、入力文字がアルファベットかどうかをチェックするプログラムの例を示します。アルファベットの場合は、大文字か小文字かをチェックします。いずれの場合にも、それぞれに応じたメッセージが表示されます。

```
// keyword_boolean_2.jsl
// Character Tester
public class BoolTest
{
    public static void main(String[] args)
```

```
    {
        try
        {
            System.out.print("Enter a character: ");
            char c = (char)System.in.read();

            if (Character.isLetter(c))
                if (Character.isLowerCase(c))
                    System.out.println("The character is lowercase.");
                else
                    System.out.println("The character is uppercase.");
            else
                System.out.println("The character is not an alphabetic character.");
        }
        catch (java.io.IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

入力

```
X
```

出力例

```
Enter a character: X
The character is uppercase.
```

他の実行例は次のようになります。

```
Enter a character: x
The character is lowercase.

Enter a character: 2
The character is not an alphabetic character.
```

参照

関連項目

J# のキーワード
false (Visual J#)
if (Visual J#)
int (Visual J#)
true (Visual J#)

その他の技術情報

プリミティブ型 (Visual J#)

byte (Visual J#)

byte キーワードは、次の表に示されたサイズと範囲に従って値を格納する整数型を示します。

型	範囲	サイズ
byte	-128 ～ 127	符号付き 8 ビット整数

リテラル

byte 変数の宣言と初期化の例を次に示します。

```
byte myByte = 127;
```

上記のように宣言すると、整数リテラル 127 は暗黙的に **int** から **byte** に変換されます。整数リテラルが **byte** の範囲を超えると、コンパイル エラーになります。

オーバーロードされたメソッドを呼び出す場合は、キャストを使用する必要があります。たとえば、**byte** パラメータと **int** パラメータを使用したオーバーロードされたメソッドがあるとします。

```
public static void myMethod(int i) {}
public static void myMethod(byte b) {}
```

byte キャストを使用すると、正しい型が呼び出されます。次に例を示します。

```
    // Calling the method with the int parameter.
MyClass.myMethod(5);
    // Calling the method with the byte parameter.
MyClass.myMethod((byte)5);
```

変換

byte から **short**、**int**、**long**、**float**、**double** への暗黙の型変換が組み込まれています。

より大きな記憶領域を持つ、リテラル以外の数値型を暗黙的に **byte** に変換することはできません。たとえば、2 つの **byte** 変数 `x` と `y` があるとします。

```
byte x = 10, y = 20;
```

次の代入ステートメントは、代入演算子の右側にある算術式が既定で **int** に評価されるため、コンパイル エラーになります。

```
// Error: conversion from int to byte:
byte z = x + y;
```

このエラーを修正するには、キャストを使用します。

```
// OK: explicit conversion:
byte z = (byte)(x + y);
```

ただし、次のステートメントは使用できます。このステートメントでは、変換先の変数の記憶サイズは元のサイズ以上になります。

```
byte x = 10, y = 20;
int m = x + y;
long n = x + y;
```

また、浮動小数点型から **byte** への暗黙の型変換が行われないことに注意してください。たとえば、次のステートメントは、明示的なキャストを使用しない場合、コンパイラ エラーになります。

```
// Error: no implicit conversion from double:
```

```
byte x = 3.0;
// OK: explicit conversion:
byte y = (byte)3.0;
```

浮動小数点型と整数型の混在する算術式の詳細については、「float (Visual J#)」と「double (Visual J#)」を参照してください。

参照

関連項目

J# のキーワード
double (Visual J#)
float (Visual J#)
int (Visual J#)
long (Visual J#)
short (Visual J#)
ubyte (Visual J#)

その他の技術情報

プリミティブ型 (Visual J#)

char (Visual J#)

char キーワードを使用して、次の表に示された範囲の Unicode 文字を宣言します。Unicode 文字は、世界中の文字言語のほとんどを 16 ビット文字で表します。

型	範囲	サイズ
char	'\u0000' 〜 '\uffff'	Unicode 16 ビット文字

リテラル

char 型の定数は、文字リテラル、16 進のエスケープ シーケンス、Unicode 表現として記述できます。また、整数の文字コードをキャストできます。次のステートメントはすべて **char** 変数を宣言し、文字 `X` を使って初期化しています。

```
// Character literal:
char myChar = 'X';
// Cast from integral type:
char myChar = (char)88;
// Unicode:
char myChar = '\u0058';
```

変換

char は、暗黙的に **short**、**int**、**long**、**float**、**double** のいずれかに変換されます。ただし、他の型から **char** 型への暗黙の型変換はありません。

参照

関連項目

J# のキーワード
double (Visual J#)
float (Visual J#)
int (Visual J#)
long (Visual J#)
short (Visual J#)

その他の技術情報

プリミティブ型 (Visual J#)

double (Visual J#)

double キーワードは、64 ビットの浮動小数点数を格納する単純型を示します。**double** 型の有効桁数とおおよその範囲は、次のとおりです。

型	おおよその範囲	精度
double	-1.7E308 ～ 1.7E308	15 ～ 16 桁

リテラル

既定では、代入演算子の右側にある実数値リテラルは **double** として扱われます。ただし、整数を **double** として扱う必要がある場合は、サフィックス d または D を使用します。次に例を示します。

```
double x = 3D;
```

変換

数値の整数型と浮動小数点型を 1 つの式に混在させることができます。混在する場合は、整数型が浮動小数点型に変換されます。式の評価は、次の規則に従います。

- 浮動小数点型のうちの 1 つが **double** である場合、式は **double** になります。関係式またはブール式の場合は **boolean** になります。
- 式に **double** 型が 1 つも存在しない場合、式は **float** になります。関係式またはブール式の場合は **boolean** になります。

浮動小数点の式に含まれる値は、次のとおりです。

- 正および負の 0
- 正および負の無限大
- 非数 (NaN)
- 有限の 0 以外の値

値の詳細については、「IEEE Standard for Binary Floating-Point Arithmetic」(http://www.ieee.org/portal/index.jsp) を参照してください。

例 1

次の例では、**int**、**short**、**float**、および **double** を加算した結果が、**double** 型として出力されます。

```
// keyword_double.jsl
// Mixing types in expressions
public class MixedTypes
{
    public static void main(String[] args)
    {
        int x = 3;
        float y = 4.5f;
        short z = 5;
        double w = 1.7E+3;
    // Double result.
        System.out.println("The sum is " + (x + y + z + w));
    }
}
```

出力

```
The sum is 1712.5
```

参照

関連項目

J# のキーワード

boolean (Visual J#)

float (Visual J#)
int (Visual J#)
short (Visual J#)
その他の技術情報
プリミティブ型 (Visual J#)

float (Visual J#)

float キーワードは、32 ビットの浮動小数点数を格納する単純型を示します。**float** 型の有効桁数とおおよその範囲は、次のとおりです。

型	おおよその範囲	精度
float	-3.4E38 ～ 3.4E38	7 桁

リテラル

既定では、代入演算子の右側にある実数値リテラルは **double** として扱われます。したがって、float 変数を初期化するには、サフィックス `f` または `F` を使用します。次に例を示します。

```
float x = 3.5F;
```

上の宣言でサフィックスを使用しなかった場合は、**double** 値を **float** 変数に格納することになるため、コンパイル エラーになります。

変換

数値の整数型と浮動小数点型を 1 つの式に混在させることができます。混在する場合は、整数型が浮動小数点型に変換されます。式の評価は、次の規則に従います。

- 浮動小数点型のうちの 1 つが **double** である場合、式は **double** になります。関係式またはブール式の場合は **boolean** になります。
- 式に **double** 型が 1 つも存在しない場合、式は **float** になります。関係式またはブール式の場合は **boolean** になります。

浮動小数点の式に含まれる値は、次のとおりです。

- 正および負の 0
- 正および負の無限大
- 非数 (NaN)
- 有限の 0 以外の値

値の詳細については、「IEEE Standard for Binary Floating-Point Arithmetic」(http://www.ieee.org/) を参照してください。

例 1

次の例では、**int**、**short**、および **float** は数式で使用されているため、**float** 型を結果として返します。この式には **double** が存在しない点に注意してください。

```
// keyword_float.jsl
// Mixing types in expressions
class MixedTypes
{
    public static void main(String [] args)
    {
        int x = 3;
        float y = 4.5f;
        short z = 5;
        System.out.println("The result is " + (x*y/z));
    }
}
```

出力

```
The result is 2.7
```

参照

関連項目

J# のキーワード
boolean (Visual J#)
double (Visual J#)

int (Visual J#)
short (Visual J#)
その他の技術情報
プリミティブ型 (Visual J#)

int (Visual J#)

int キーワードは、次の表に示されたサイズと範囲に従って値を格納する整数型を示します。

型	範囲	サイズ
int	-2,147,483,648 ～ 2,147,483,647	符号付き 32 ビット整数

リテラル

int 型の変数は、次のように宣言および初期化できます。

```
int myInt = 123;
```

サフィックスがない整数リテラルの型は、**int** および **long** のうち、その整数の値を表すことができる最も範囲の狭い型になります。上の例では、**int** 型です。

変換

int から **long**、**float**、または **double** への暗黙の型変換が組み込まれています。次に例を示します。

```
// OK: implicit conversion to float
float myFloat = 123;
```

byte、**ubyte**、**short**、UInt16、**char** から **int** への暗黙の型変換が組み込まれています。たとえば、**long** 変数 `myLong` を使用する場合、キャストなしでは次の代入ステートメントはコンパイル エラーになります。

```
long myLong = 22;
// Error: no implicit conversion from long.
int myInt = myLong;
// OK: explicit conversion.
int myInt = (int)myLong;
```

浮動小数点型から **int** への暗黙の型変換はありません。たとえば、次のステートメントは、明示的なキャストを使用しない場合、コンパイラ エラーになります。

```
// Error: no implicit conversion from double.
int x = 3.0;
// OK: explicit conversion.
int y = (int)3.0;
```

浮動小数点型と整数型の混在する算術式の詳細については、「float (Visual J#)」と「double (Visual J#)」を参照してください。

参照

関連項目

J# のキーワード
byte (Visual J#)
char (Visual J#)
double (Visual J#)
float (Visual J#)
long (Visual J#)
short (Visual J#)
ubyte (Visual J#)

その他の技術情報

プリミティブ型 (Visual J#)

long (Visual J#)

long キーワードは、次の表に示されたサイズと範囲に従って値を格納する整数型を示します。

型	範囲	サイズ
long	-9,223,372,036,854,775,808 ～ 9,223,372,036,854,775,807	符号付き 64 ビット整数

リテラル

long 変数の宣言と初期化の例を次に示します。

```
long myLong = 429496729;
```

上記のように宣言すると、整数リテラル 429496729 は暗黙的に **int** から **long** に変換されます。整数リテラルを **long** の格納場所に格納できない場合は、コンパイル エラーになります。

また、**long** 型にサフィックス L を付けることもできます。次に例を示します。

```
long myLong = 4294967296L;
```

サフィックス L を付けたリテラル整数は **long** として扱われます。

サフィックスは、オーバーロードされたメソッドの呼び出しでよく使われます。たとえば、**long** パラメータと **int** パラメータを使用したオーバーロードされたメソッドがあるとします。

```
public static void myMethod(int i) {}
public static void myMethod(long l) {}
```

サフィックス L を使用すると、正しい型が呼び出されます。次に例を示します。

```
// Calling the method with the int parameter:
MyClass.myMethod(5);
// Calling the method with the long parameter:
MyClass.myMethod(5L);
```

long 型とその他の数値の整数型を同じ式で使用できます。その場合、式は **long** として評価されます。関係式またはブール式の場合は **boolean** として評価されます。**long** として評価される式の例を次に示します。

```
898L + 88
```

メモ :

小文字の "l" もサフィックスとして使用できます。ただし、小文字の "l" は数字の "1" と間違えやすいので注意が必要です。明確にするには "L" を使用します。

浮動小数点型と整数型の混在する算術式の詳細については、「float (Visual J#)」と「double (Visual J#)」を参照してください。

変換

long から **float** または **double** への暗黙の型変換が組み込まれています。その他の型の場合は、キャストを使用する必要があります。たとえば、次のステートメントは、明示的なキャストを使用しない場合、コンパイル エラーになります。

```
// Error: no implicit conversion from long to int:
int x = 8L;
// OK: explicit conversion to int:
int x = (int)8L;
```

byte、**ubyte**、**short**、**int**、**char** から **long** への暗黙の型変換が組み込まれています。

また、浮動小数点型から **long** への暗黙の型変換が行われないことに注意してください。たとえば、次のステートメントは、明示的なキャストを使用しない場合、コンパイラ エラーになります。

```
// Error: no implicit conversion from double:
long x = 3.0;
// OK: explicit conversion:
long y = (long)3.0;
```

参照

関連項目

J# のキーワード
byte (Visual J#)
char (Visual J#)
double (Visual J#)
boolean (Visual J#)
float (Visual J#)
int (Visual J#)
short (Visual J#)
ubyte (Visual J#)

その他の技術情報

プリミティブ型 (Visual J#)

short (Visual J#)

short キーワードは、次の表に示されたサイズと範囲に従って値を格納する整数データ型を示します。

型	範囲	サイズ
short	-32,768 ～ 32,767	符号付き 16 ビット整数

リテラル

short 変数の宣言と初期化の例を次に示します。

```
short x = 32767;
```

上のように宣言すると、整数リテラル `32767` は **int** から **short** に暗黙的に変換されます。整数リテラルを **short** の格納場所に格納できない場合は、コンパイル エラーになります。

オーバーロードされたメソッドを呼び出す場合は、キャストを使用する必要があります。たとえば、**short** パラメータと **int** パラメータを使用したオーバーロードされたメソッドがあるとします。

```
public static void myMethod(int i) {}
public static void myMethod(short s) {}
```

short キャストを使用すると、正しい型が呼び出されます。次に例を示します。

```
// Calling the method with the int parameter:
MyClass.myMethod(5);
// Calling the method with the short parameter:
MyClass.myMethod((short)5);
```

変換

short から **int**、**long**、**float**、**double** への暗黙の型変換が組み込まれています。

より大きな記憶領域を持つ、リテラル以外の数値型を暗黙的に **short** に変換することはできません。たとえば、2 つの **short** 変数 `x` と `y` があるとします。

```
short x = 5, y = 12;
```

次の代入ステートメントは、代入演算子の右側にある算術式が既定で **int** に評価されるため、コンパイル エラーになります。

```
// Error: no conversion from int to short:
short z = x + y;
```

このエラーを修正するには、キャストを使用します。

```
// OK: explicit conversion:
short z = (short)(x + y);
```

ただし、次のステートメントは使用できます。このステートメントでは、変換先の変数の記憶サイズは元のサイズ以上になります。

```
int m = x + y;
long n = x + y;
```

浮動小数点型から **short** への暗黙の型変換はありません。たとえば、次のステートメントは、明示的なキャストを使用しない場合、コンパイラ エラーになります。

```
// Error: no implicit conversion from double:
short x = 3.0;
```

```
    // OK: explicit conversion:
short y = (short)3.0;
```

浮動小数点型と整数型の混在する算術式の詳細については、「float (Visual J#)」と「double (Visual J#)」を参照してください。

参照

関連項目

J# のキーワード

double (Visual J#)

float (Visual J#)

int (Visual J#)

long (Visual J#)

その他の技術情報

プリミティブ型 (Visual J#)

ubyte (Visual J#)

ubyte キーワードは、次の表に示された値を格納する整数型を示します。

型	範囲	サイズ
ubyte	0 ～ 255	符号なし 8 ビット整数

リテラル

ubyte 変数の宣言と初期化の例を次に示します。

```
ubyte myByte = 255;
```

上のように宣言すると、整数リテラル `255` は **int** から **ubyte** に暗黙的に変換されます。整数リテラルが **ubyte** の範囲を超えると、コンパイル エラーになります。

変換

ubyte から **short**、**int**、**long**、**float**、**double** への暗黙の型変換が組み込まれています。

より大きな記憶領域を持つ、リテラル以外の数値型を暗黙的に **ubyte** に変換することはできません。たとえば、2 つの **ubyte** 変数 `x` と `y` があるとします。

```
ubyte x = 10, y = 20;
```

次の代入ステートメントは、代入演算子の右側にある算術式が既定で **int** に評価されるため、コンパイル エラーになります。

```
// Error: conversion from int to ubyte:
ubyte z = x + y;
```

このエラーを修正するには、キャストを使用します。

```
// OK: explicit conversion:
ubyte z = (ubyte)(x + y);
```

ただし、次のステートメントは使用できます。このステートメントでは、変換先の変数の記憶サイズは元のサイズ以上になります。

```
int x = 10, y = 20;
int m = x + y;
long n = x + y;
```

浮動小数点型から **ubyte** への暗黙の型変換はありません。たとえば、次のステートメントは、明示的なキャストを使用しない場合、コンパイラ エラーになります。

```
// Error: no implicit conversion from double
ubyte x = 3.0;
// OK: explicit conversion:
ubyte y = (ubyte)3.0;
```

オーバーロードされたメソッドを呼び出すときは、キャストを使用する必要があります。たとえば、**ubyte** パラメータと **int** パラメータを使用したオーバーロードされたメソッドがあるとします。

```
public static void myMethod(int i) {}
public static void myMethod(ubyte b) {}
```

ubyte キャストを使用すると、正しい型が呼び出されます。次に例を示します。

```
// Calling the method with the int parameter.
```

```
MyClass.myMethod(5);
// Calling the method with the ubyte parameter.
MyClass.myMethod((ubyte)5);
```

浮動小数点型と整数型の混在する算術式の詳細については、「float (Visual J#)」と「double (Visual J#)」を参照してください。

参照

関連項目

J# のキーワード
byte (Visual J#)
double (Visual J#)
float (Visual J#)
int (Visual J#)
long (Visual J#)
short (Visual J#)

その他の技術情報

プリミティブ型 (Visual J#)

選択ステートメント (Visual J#)

選択ステートメントは、指定された条件が true か false かに基づいて、プログラムの制御を特定の流れに移動させます。

選択ステートメントでは、次のキーワードを使用します。

if (Visual J#)

switch (Visual J#)

参照

関連項目

J# のキーワード

if (Visual J#)

if ステートメントは、ブール型の式の値に基づいて、実行するステートメントを選択します。形式は次のとおりです。

```
if (expression)
      statement1
[else
      statement2]
```

パラメータ

expression

暗黙的に **boolean** に変換できる式。

statement1

expression が **true** の場合に実行される埋め込みステートメント。

statement2

expression が **false** の場合に実行される埋め込みステートメント。

解説

expression が **true** の場合、*statement1* が実行されます。オプションの **else** 句が存在し、*expression* が **false** に評価された場合、*statement2* が実行されます。**if** ステートメントを実行した後、制御は次のステートメントに移動します。

if ステートメントの結果 (**true** または **false**) で複数のステートメントを実行する場合、複数のステートメントをブロック内に含めることによって、複数のステートメントを条件に応じて実行できます。

条件の評価に従って実行されるステートメントは、元の **if** ステートメントに入れ子になった、他の **if** ステートメントなど、どのような種類のステートメントでもかまいません。**if** ステートメントが入れ子にされている場合、**else** 句は、対応する **else** を持たない最後の **if** に従属します。次に例を示します。

```
if (x > 10)
   if (y > 20)
      System.out.println("Statement_1");
   else
      System.out.println("Statement_2");
```

この例では、`Statement_2` は、条件 `(y > 20)` が **false** の場合に表示されます。ただし、`Statement_2` と条件 `(x >10)` を関連付ける場合は次のように中かっこを使用します。

```
if (x > 10)
{
   if (y > 20)
      System.out.println("Statement_1");
}
else
   System.out.println("Statement_2");
```

この場合は、`Statement_2` は、条件 `(x > 10)` が **false** の場合に表示されます。

例 1

次の例では、**for** ループを使用し、0 から 5 までの数値について、それぞれ奇数であるか偶数であるかをチェックして結果を表示します。

```
// if-else1.jsl
// if-else example

public class MyClass
{
    public static void main(String[] args)
    {
```

```
        for (int i = 0; i <= 5; i++)
        {
            if (Math.IEEEremainder(i, 2.0) == 0)
                System.out.println(i + " even");
            else
                System.out.println(i + " odd");
        }
    }
}
```

出力

```
0 even
1 odd
2 even
3 odd
4 even
5 odd
```

また、次のように **else-if** を使用することによって **if** ステートメントを拡張し、複数の条件を処理することもできます。

```
if (Condition_1)
   Statement_1;
else if (Condition_2)
   Statement_2;
else if (Condition_3)
   Statement_3;
...
else
   Statement_n;
```

例 2

入力文字が小文字、大文字、数字のいずれであるかをチェックする例を次に示します。それ以外の場合、入力文字は英数字ではありません。このプログラムでは、**else-if** による条件分岐を使用します。

```
// if-else2.jsl
// if example

public class IfTest
{
    public static void main(String[] args)
    {
        try
        {
        System.out.print("Enter a character: ");
        int i = System.in.read();
        char c = (char)i;

        if (Character.isUpperCase(c))
            System.out.println("The character is uppercase.");
        else if (Character.isLowerCase(c))
            System.out.println("The character is lowercase.");
        else if (Character.isDigit(c))
            System.out.println("The character is a number.");
        else
            System.out.println("The character is not alphanumeric.");
        }
        catch (Exception e)
        {
            // You may insert a catch statement or do nothing.
        }
    }
}
```

入力

```
E
```

出力例

```
Enter a character: E
The character is uppercase.
```

他の実行例は次のようになります。

実行サンプル 2 :

```
Enter a character: e
The character is lowercase.
```

実行サンプル 3 :

```
Enter a character: 4
The character is a number.
```

実行サンプル 4 :

```
Enter a character: $
The character is not alphanumeric.
```

参照

関連項目

J# のキーワード

switch (Visual J#)

switch (Visual J#)

switch ステートメントは、ステートメントの本体にある **case** ステートメントの 1 つに制御を渡すことで複数選択を処理する制御ステートメントです。形式は次のとおりです。

```
switch (expression)
{
   case constant-expression:
            statement
            jump-statement
   [default:
            statement
            jump-statement]
}
```

パラメータ

expression

序数型 (**integral type** または **char or enum**) の式。

statement

制御が **case** または **default** に移ったときに実行される埋め込みステートメント。

jump-statement

case 本体の外部に制御を移動させる **break** ステートメント。

constant-expression

制御は、この式の値に従って特定の **case** に移動します。

解説

プログラムの制御は、*constant-expression* が *expression* と一致する **case** ステートメントに移動します。また **case** 節はいくつ指定してもかまいませんが、定数式の値は、**switch** ステートメント内で重複できません。ステートメント本体の実行は指定されたステートメントから始まり、*jump-statement* によって制御が **case** 本体の外部に移動するまで実行されます。

各 **case** ブロックの後には、*jump-statement* パラメータが必要です。C++ の **switch** ステートメントとは異なり、J# では、case ステートメントが空である (コードが存在しない) 場合を除き、ある case ラベルからその下の case ラベルへの明示的な落下 (フォール スルー) をサポートしていません。

expression がいずれの *constant-expression* とも一致しない場合、制御はオプションの **default** ラベルに続く *statement* または *statements* に移動します。**default** ラベルがない場合、制御は **switch** の外部に移動します。

例 1

```
// switch1.jsl
// swicth example
import java.io.*;

class SwitchTest
{
    public static void main(String[] args)
    {
        System.out.println("Coffee sizes: 1=Small 2=Medium 3=Large");
        System.out.print("Please enter your selection: ");
        int cost = 0;
        try
        {
            char n = (char)System.in.read();

            switch (n)
            {
                case '1':
                    cost = 25;
```

```
                    break;
                case '2':
                    cost = 50;
                    break;
                case '3':
                    cost = 75;
                    break;
                default:
                    System.out.println
                    ("Invalid selection. Please select 1, 2, or 3.");
            }
            if (cost != 0)
                System.out.println("Please insert " + cost +
                                   " cents.");
            System.out.println("Thank you for your business.");
        }
        catch (IOException e)
        {
            // You may insert a catch statement or do nothing.
        }
    }
}
```

入力

```
2
```

出力例

```
Coffee sizes: 1=Small 2=Medium 3=Large
Please enter your selection: 2
Please insert 50 cents.
Thank you for your business.
```

コードの説明

前述の例では、**char** 型の変数 n が **switch** の case 節に使用されています。また、**int** などの序数変数を使用することもできます。その場合は、**switch** の case 節を次のように使用します。

```
switch(s)
{
   case 1:
      ...
   case 2:
      ...
}
```

例 2

次のサンプルでは、空の case ラベルについて、ある case ラベルからその下の case ラベルへの落下 (フォール スルー) を許可する例を示します。

```
// switch2.jsl
// switch example

class SwitchTest
{
    public static void main(String[] args)
    {
        int n = 1;
        switch(n)
        {
            case 1:
            case 2:
            case 3:
                 System.out.println("It's 3.");
                 break;
            default:
```

```
                System.out.println("Not sure what it is.");
            }
        }
    }
```

出力

```
It's 3.
```

参照

関連項目

J# のキーワード

if (Visual J#)

参照型 (Visual J#)

ここでは、参照型の宣言に使用するキーワードについて説明します。キーワードは、次のとおりです。

- class (Visual J#)
- delegate (Visual J#)
- interface (Visual J#)

参照

関連項目

J# のキーワード

class (Visual J#)

クラスは **class** キーワードを使用し、次の形式で宣言します。

```
[attributes] [modifiers] class identifier [extends super-class] [implements interface-list]
{ class-body }[;]
```

パラメータ

attributes (省略可能)

追加の宣言情報。

modifiers (省略可能)

使用できる修飾子は、abstract、final、private、protected、public、static、strictfp です。

identifier

クラス名。

super-class (省略可能)

このクラスが継承する基本クラス。

interface-list (省略可能)

実装するインターフェイス。複数指定する場合は、コンマで区切ります。

class-body

クラス メンバの宣言。

解説

J# では単一継承のみ使用できます。つまり、1 つのクラスが継承できるスーパー クラスは 1 つだけです。ただし、クラスは複数のインターフェイスを実装できます。クラスの継承とインターフェイスの実装の例を次の表に示します。

継承	例
なし	`class ClassA { }`
1 つ	`class DerivedClass extends BaseClass { }`
なし。2 つのインターフェイスを実装。	`class ImplClass implements IFace1, IFace2 { }`
1 つ。1 つのインターフェイスを実装。	`class ImplDerivedClass extends BaseClass implements IFace1 { }`

アクセス レベルの protected (Visual J#) と private (Visual J#) は、入れ子になったクラスだけで使用できます。

クラスには、メンバ メソッドおよびメンバ フィールド (インスタンス変数) の宣言のみ含めることができます。また、クラスに、複数のクラスやインターフェイスを含めることもできます。

例 1

ここでは、クラスのフィールド、コンストラクタ、メソッドの宣言例を示します。オブジェクト インスタンスの作成、インスタンス データの出力の例も示します。この例では、2 つのクラスを宣言します。`Kid` クラスには、2 つのプライベート フィールド (`name` および `age`,) と、2 つのパブリック メソッドがあります。2 番目のクラスはデータ処理のための `MainClass` です。

```
// keyword_class.jsl
// class example

public class Kid
{
    private int age;
    private String name;
```

```
    // Default constructor:
    public Kid()
    {
        name = "N/A";
    }

    // Constructor:
    public Kid(String name, int age)
    {
        this.name = name;
        this.age = age;
    }

    // Printing method:
    public void PrintKid()
   {
        System.out.println(name + ", " + age + " years old.");
    }
}

public class MainClass
{
    public static void main(String[] args)
    {
        // Create objects.
        // Objects must be created using the new operator:
        Kid kid1 = new Kid("Isabella", 4);
        Kid kid2 = new Kid("Angelina", 2);

        // Create an object using the default constructor:
        Kid kid3 = new Kid();

        // Display results:
        System.out.print("Kid #1: ");
        kid1.PrintKid();
        System.out.print("Kid #2: ");
        kid2.PrintKid();
        System.out.print("Kid #3: ");
        kid3.PrintKid();
    }
}
```

出力

```
Kid #1: Isabella, 4 years old.
Kid #2: Angelina, 2 years old.
Kid #3: N/A, 0 years old.
```

上記の例で、プライベート フィールド (`name` および `age`) にアクセスできるのは、`Kid` クラスのパブリック メソッドだけであることに注意してください。たとえば、次のステートメントを使用して `Main` メソッドから子供の名前を出力できません。

```
// Error
System.out.print(kid1.name);
```

ただし、`Main` メソッドが `Kid` クラスのメンバであった場合は出力できます。

また、既定のコンストラクタを使って作成したオブジェクト (`kid3`) は、年齢フィールドが 0 に初期化されることに注意してください。

参照

関連項目

J# のキーワード

interface (Visual J#)

delegate (Visual J#)

delegate キーワードは、指定されたメソッドをカプセル化するための参照型を宣言する場合に使用します。デリゲートは C++ の関数ポインタとほぼ同じですが、タイプ セーフであり、安全です。

delegate 型の宣言は、次の形式をとります。

```
[attributes] [modifiers] delegate result-type identifier ([formal-parameters]);
```

パラメータ

attributes (省略可能)

追加の宣言情報。

modifiers (省略可能)

public、**private**、**protected** の各アクセス修飾子を使用できます。

result-type

結果の型。指定したメソッドに対してデリゲートを関連付ける場合は、結果の型と、メソッドの戻り値の型とを一致させる必要があります。

identifier

デリゲート名。

formal-parameters (省略可能)

パラメータ リスト。

解説

delegate を使用すると、関数をパラメータとして渡すことができます。デリゲートはタイプ セーフなので、**delegate** として渡す関数には、**delegate** 宣言と同じシグネチャが必要です。

デリゲートは、イベントの基礎になります。

例 1

delegate の宣言と使用について、簡単な例を次に示します。

```
// keyword_delegate_1.jsl
// delegate declaration
/** @delegate */
delegate void MyDelegate(int i);

class Program
{
    public static void main(String[] args)
    {
        Program p = new Program();
        MyDelegate d = new MyDelegate(p.DelegateFunction);
        p.TakesADelegate(d);
    }

    public void TakesADelegate(MyDelegate SomeFunction)
    {
        SomeFunction.Invoke(21);
    }

    public void DelegateFunction(int i)
    {
        System.out.println(
            "Called by delegate with number: " + i + ".");
    }
}
```

出力

```
Called by delegate with number: 21.
```

例 2

次の例では、1 つのデリゲートを **static** メソッドと **instance** メソッドの両方に割り当て、各メソッドから特定の情報を戻します。

```
// keyword_delegate_2.jsl
// Calling both static and instance methods from delegates

// Delegate declaration.
/** @delegate */
delegate void MyDelegate();

public class MyClass
{
    public void InstanceMethod()
    {
        System.out.println("A message from the instance method.");
    }

    static public void StaticMethod()
    {
        System.out.println("A message from the static method.");
    }
}

public class MainClass
{
    static public void main(String[] args)
    {
        MyClass p = new MyClass();

        // Map the delegate to the instance method:
        MyDelegate d = new MyDelegate(p.InstanceMethod);
        d.Invoke();

        // Map to the static method:
        d = new MyDelegate(MyClass.StaticMethod);
        d.Invoke();
    }
}
```

出力

```
A message from the instance method.
A message from the static method.
```

参照

関連項目

J# のキーワード
private (Visual J#)
protected (Visual J#)
public (Visual J#)

その他の技術情報

参照型 (Visual J#)

interface (Visual J#)

interface はコントラクトを定義します。**interface** を実装するクラスは、そのコントラクトに従う必要があります。次の形式で宣言します。

```
[attributes] [access-modifier] interface identifier [extends interface-list] {interface-bod
y}
```

パラメータ

attributes (省略可能)

追加の宣言情報。

access-modifier (省略可能)

使用できる修飾子は **public**、**protected**、**private**、**abstract**、**static**、**strictfp** です。

identifier

インターフェイスの名前。

interface-list (省略可能)

インターフェイスの継承元となる外部のインターフェイス (複数指定する場合はコンマ区切り)。

interface-body

インターフェイス メンバの宣言。

解説

interface には、名前空間またはクラスを指定できます。また、メソッドやフィールド (変数) を含めることもできます。

インターフェイスのメソッドは、暗黙的に public および abstract になります。

インターフェイスのフィールドは必ず初期化する必要があります。暗黙的に public、static、final になります。

1 つ以上の基本インターフェイスを拡張するインターフェイスを作成できます。2 つの基本インターフェイス `IBase1` と `IBase2` を拡張して、`IMyInterface` インターフェイスを作成する例を次に示します。

```
interface IMyInterface extends IBase1, IBase2
{
    void MethodA();
    void MethodB();
}
```

インターフェイスは、クラスで実装できます。1 つのクラスで複数のインターフェイスを実装できます。次に例を示します。

```
class Class1 implements Iface1, Iface2
{
    // Class members.
}
```

インターフェイス オブジェクトを宣言することにより、そのインターフェイスを実装するオブジェクトを参照することもできます。次の 2 番目の例を参照してください。

例 1

ここでは、インターフェイスの実装例を示します。次の例では、`IPoint` インターフェイスには、複数のフィールド宣言と、点オブジェクトを表示するメソッドが 1 つ存在します。

```
// keyword_interface1.jsl
// Interface implementation

interface IPoint
{
```

```
    int x = 0;
    int y = 0;
    void PrintPoint();
}

class MyPoint implements IPoint
{
    // Fields (variables):
    int x;
    int y;

    // Constructor:
    public MyPoint(int x, int y)
    {
      this.x = x;
      this.y = y;
    }
    // An extra method that does not exist in the interface:
    public void PrintPoint()
    {
        System.out.println("x=" + x + ", y="+ y);
    }
}

class MainClass
{
    public static void main(String[] args)
    {
        MyPoint p = new MyPoint(2,3);
        System.out.print("My Point using a class object: ");
        p.PrintPoint();    // Using a class object.

        // Creating an interface reference:
        System.out.print("My Point using an interface object: ");
        IPoint ip = new MyPoint(20,30);
    // Using an interface object.
        ip.PrintPoint();
    }
}
```

出力

```
My Point using a class object: x=2, y=3
My Point using an interface object: x=20, y=30
```

参照

関連項目

J# のキーワード

class (Visual J#)

アクセス キーワード (Visual J#)

ここでは、以下のアクセス キーワードについて説明します。

- super (Visual J#)

  基本クラスのメンバにアクセスします。

- this (Visual J#)

  クラスの現在のインスタンスを参照します。

参照

関連項目

J# のキーワード

super (Visual J#)

super キーワードは、派生クラス (サブクラス) 内で基本クラス (スーパー クラス) のメンバにアクセスするために使用されます。**super** キーワードを使用すると、次のことができます。

- 他のメソッドによってオーバーライドされた基本クラスのメソッドを呼び出します。
- 派生クラスのインスタンスを作成するときに呼び出される基本クラスのコンストラクタを指定します。

super キーワードを使用してスーパー クラスにアクセスできるのは、コンストラクタ内、インスタンスのメソッド内、またはインスタンスのプロパティ アクセサ内だけです。

静的メソッド内で **super** キーワードを使用するのはエラーです。

例 1

この例では、基本クラス `Person` および派生クラス `Employee` の両方に、`Getinfo` という名前のメソッドがあります。**super** キーワードを使用すると、基本クラスの `Getinfo` メソッドを派生クラス内で呼び出すことができます。

```
// super1.jsl
// Accessing base class members

    public class Person
    {
        protected String ssn = "555-55-5555";
        protected String name = "Isabella D. Abolrous";

        public void GetInfo()
        {
            System.out.println("Name: " + name);
            System.out.println("SSN: " + ssn);
        }
    }
    class Employee extends Person
    {
        public String id = "ABC123EFG";

        public void GetInfo()
        {
            // Calling the base class GetInfo method:
            super.GetInfo();
            System.out.println("Employee ID: " + id);
        }
    }

class TestClass
{
    public static void main()
    {
        Employee E = new Employee();
        E.GetInfo();
    }
}
```

出力

```
Name: Isabella D. Abolrous
SSN: 555-55-5555
Employee ID: ABC123EFG
```

例 2

この例では、派生クラスのインスタンスを作成するときに呼び出される、基本クラスのコンストラクタを指定する方法を示します。

```
// super2.jsl
// super keyword example
```

```
public class MyBase
{
    int num;
    public MyBase()
    {
        System.out.println("I am now in MyBase().");
    }
    public MyBase(int i)
    {
        num = i;
        System.out.println("I am now in MyBase(int i), and num = "
                           + i);
    }
}

public class MyDerived extends MyBase
{
    public MyDerived()
    {
        // This constructor will call MyBase.MyBase().
        super();
    }

    public MyDerived(int i)
    {
        // This constructor will call MyBase.MyBase(int i).
        super(i);
    }

    public static void main()
    {
        MyDerived md = new MyDerived();
        MyDerived md1 = new MyDerived(555);
    }
}
```

出力

```
I am now in MyBase().
I am now in MyBase(int i), and num = 555
```

参照

関連項目

J# のキーワード

this (Visual J#)

this (Visual J#)

this キーワードは、クラスの現在のインスタンスを参照します。静的メンバ関数は、**this** ポインタを持ちません。**this** キーワードは、コンストラクタ内、インスタンスのメソッド内、およびインスタンスのアクセサ内でメンバにアクセスするために使用できます。

this の一般的な使い方を次に示します。

- 似た名前によって隠ぺいされるメンバを修飾します。たとえば、次のように使います。

```
public Employee(String name, String alias)
{
    // Use this to qualify the fields, name and alias:
    this.name = name;
    this.alias = alias;
}
```

- オブジェクトを他のメソッドにパラメータとして渡します。たとえば、次のように使います。

```
Tax.CalcTax(this)
```

静的メソッド、静的プロパティ アクセサ、またはフィールド宣言の変数初期化子で **this** を参照するのはエラーです。

例 1

この例では、似た名前によって隠ぺいされている `Employee` クラスのメンバ `name` と `alias` を修飾するために **this** が使用されています。また、別のクラスに属するメソッド `CalcTax` にオブジェクトを渡すためにも使用されています。

```
// keywords_this.jsl
// this example
import java.math.BigDecimal;
public class Employee
{
    public String name;
    public String alias;
    public BigDecimal salary = new BigDecimal(3000.00);

    // Constructor:
    public Employee(String name, String alias)
    {
        // Use this to qualify the fields, name and alias:
        this.name = name;
        this.alias = alias;
    }

    // Printing method:
    public void printEmployee()
    {
        System.out.println("Name: " + name + "\nAlias: " + alias);
        // Passing the object to the CalcTax method by using this:
        System.out.println("Taxes: $" + Tax.CalcTax(this).setScale(2));
    }
}

public class Tax
{
    public static BigDecimal CalcTax(Employee E)
    {
        return new BigDecimal(0.08 * E.salary.doubleValue());
    }
}

public class MainClass
```

```
{
    public static void main(String[] args)
    {
        // Create objects:
        Employee E1 = new Employee("John M. Trainer", "jtrainer");

        // Display results:
        E1.printEmployee();
    }
}
```

出力

```
Name: John M. Trainer
Alias: jtrainer
Taxes: $240.00
```

その他の例については、「class (Visual J#)」を参照してください。

参照

関連項目

J# のキーワード

super (Visual J#)

その他の技術情報

アクセス修飾子 (Visual J#)

J# の演算子

J# には、多くの演算子が用意されています。演算子とは、式で実行する演算を指定する記号のことです。次の表のように、J# には一般的な算術演算子と論理演算子の他にもさまざまな演算子が組み込まれています。

演算子のカテゴリ	演算子
算術	+ 演算子 (Visual J#) - 演算子 (Visual J#) * 演算子 (Visual J#) / 演算子 (Visual J#) % 演算子 (Visual J#)
論理 (ブールおよびビットごと)	& 演算子 (Visual J#) | 演算子 (Visual J#) ^ 演算子 (Visual J#) ! 演算子 (Visual J#) ~ 演算子 (Visual J#) && 演算子 (Visual J#) || 演算子 (Visual J#)
文字列の連結	+ 演算子 (Visual J#)
インクリメント、デクリメント	++ 演算子 (Visual J#) -- 演算子 (Visual J#)
シフト	<< 演算子 (Visual J#) >> 演算子 (Visual J#) >>> 演算子 (Visual J#)
関係演算	== 演算子 (Visual J#) != 演算子 (Visual J#) < 演算子 (Visual J#) > 演算子 (Visual J#) <= 演算子 (Visual J#) >= 演算子 (Visual J#)

代入	= 演算子 (Visual J#) += 演算子 (Visual J#) -= 演算子 (Visual J#) *= 演算子 (Visual J#) /= 演算子 (Visual J#) %= 演算子 (Visual J#) &= 演算子 (Visual J#) |= 演算子 (Visual J#) != 演算子 (Visual J#) <<= 演算子 (Visual J#) >>= 演算子 (Visual J#)
メンバ アクセス	. 演算子 (Visual J#)
インデックス配列	[] 演算子 (Visual J#)
キャスト	() 演算子 (Visual J#)
条件	?: 演算子 (Visual J#)
オブジェクト作成	new 演算子 instanceof 演算子

参照

関連項目

J# のキーワード

>= 演算子 (Visual J#)

すべてのプリミティブ数値型では、"より大きいか等しい" 関係演算子 (**>=**) が定義されています。この演算子では、最初のオペランドが 2 番目のオペランドより大きいか等しい場合に **true** が返され、それ以外の場合は **false** が返されます。

```
expr1 >= expr2
```

パラメータ

expr1

プリミティブ数値型の式。

expr2

プリミティブ数値型の式。

解説

オペランドの上位変換された型が **int** または **long** のときは、符号付き整数の比較が実行され、この上位変換された型が **float** または **double** のときは、浮動小数点の比較が実行されます。

使用例

```
// operator_greater_than_or_equal.jsl

class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(1.1 >= 1);
        System.out.println(1.1 >= 1.1);
    }
}
```

出力

```
true
true
```

参照

その他の技術情報

J# の演算子

&& 演算子 (Visual J#)

条件 AND 演算子 (&&) では **boolean** オペランドの論理 AND が実行されますが、必要な場合のみ、2 番目のオペランドが評価されます。

```
expr1 && expr2
```

パラメータ

expr1

boolean 型の式。

expr2

boolean 型の式。

解説

```
x && y
```

この演算は次の演算に相当します。

```
x & y
```

ただし、`x` が `false` の場合、`y` は評価されません。この場合、AND 演算の結果は `y` の値にかかわらず `false` になるためです。これは、"ショートサーキット" 評価と呼ばれます。

使用例

最初のオペランドだけが評価される **&&** を使用した式の例は、次のとおりです。

```
// operator_logical_and.jsl

class MyClass
{
    static boolean fn1()
    {
        System.out.println("fn1 called");
        return false;
    }

    static boolean fn2()
    {
        System.out.println("fn2 called");
        return true;
    }

    public static void main(String[] args)
    {
        System.out.println("Regular AND:");
        System.out.println("Result is: " + (fn1() & fn2()));
        System.out.println("Short-circuit AND:");
        System.out.println("Result is: " + (fn1() && fn2()));
    }
}
```

出力

```
Regular AND:
fn1 called
fn2 called
Result is: false
Short-circuit AND:
```

```
fn1 called
Result is: false
```

参照

その他の技術情報

J# の演算子

% 演算子 (Visual J#)

剰余演算子 (%) では、最初のオペランドが 2 番目のオペランドで除算された後の剰余が計算されます。すべての数値型には定義済みの剰余演算子があります。

```
expr1 % expr2
```

パラメータ

expr1

プリミティブ数値型の式。

expr2

プリミティブ数値型の式。

使用例

```
// operator_modulus.jsl

class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(5 % 2);         // int
        System.out.println(-5 % 2);        // int
        System.out.println(5.0 % 2.2);     // double
        System.out.println(5.0f % 2.2f);   // float
    }
}
```

出力

```
1
-1
0.59999999999999964
0.5999999
```

参照

その他の技術情報

J# の演算子

- 演算子 (Visual J#)

- 演算子は、単項演算子または二項演算子として機能します。

```
- expr
expr1 - expr2
```

パラメータ

expr

プリミティブ数値型または列挙型の式。

expr1

プリミティブ数値型の式。

expr2

プリミティブ数値型の式。

解説

単項 - 演算子は、すべての数値型および列挙型に対してあらかじめ定義されています。数値型での単項 - 演算の結果は、オペランドの符号を反転した値になります。

二項 - 演算子はプリミティブ数値型に対してのみ適用でき、最初のオペランドから 2 番目のオペランドを減算するようにあらかじめ定義されています。

使用例

```
// operator_minus.jsl
class MyClass
{
    public static void main(String[] args)
    {
        int a = 5;
        System.out.println(-a);
        System.out.println(a - 1);
        System.out.println(a - .5);
    }
}
```

出力

```
-5
4
4.5
```

参照

その他の技術情報

J# の演算子

-- 演算子 (Visual J#)

デクリメント演算子 (--) では、オペランドが 1 ずつデクリメントされます。デクリメント演算子は、オペランドの前または後に指定できます。

```
-- var
var --
```

パラメータ

var

プリミティブ数値型の式。

解説

最初の形式は、前置デクリメント演算です。この演算の結果は、デクリメントが行われた後のオペランドの値になります。

2 番目の形式は、後置デクリメント演算です。この演算の結果は、デクリメントが行われる前のオペランドの値になります。

final として宣言されている変数は、デクリメント演算子のオペランドとして使用できません。

使用例

```
// operator_decrement.jsl

class MyClass
{
    public static void main(String[] args)
    {
        double x;
        x = 1.5;
        System.out.println(--x);
        x = 1.5;
        System.out.println(x--);
        System.out.println(x);
    }
}
```

出力

```
0.5
1.5
0.5
```

参照

その他の技術情報

J# の演算子

|| 演算子 (Visual J#)

条件 OR 演算子 (**||**) では **boolean** オペランドの論理 OR が実行されますが、必要な場合だけ、2 番目のオペランドが評価されます。

```
expr1 || expr2
```

パラメータ

expr1

boolean 型の式。

expr2

boolean 型の式。

解説

```
x || y
```

この演算は次の演算に相当します。

```
x | y
```

ただし、`x` が `true` の場合、`y` は評価されません。この場合、OR 演算の結果は `y` の値にかかわらず `true` になるためです。これは、"ショートサーキット" 評価と呼ばれます。

使用例

最初のオペランドだけが評価される **||** を使用した式の例は、次のとおりです。

```
// operator_short_circuit_OR.jsl

class MyClass
{
    static boolean fn1()
    {
        System.out.println("fn1 called");
        return true;
    }

    static boolean fn2()
    {
        System.out.println("fn2 called");
        return false;
    }

    public static void main(String[] args)
    {
        System.out.println("Regular OR:");
        System.out.println("Result is: " + (fn1() | fn2()));
        System.out.println("Short-circuit OR:");
        System.out.println("Result is: " + (fn1() || fn2()));
    }
}
```

出力

```
Regular OR:
fn1 called
fn2 called
Result is: true
Short-circuit OR:
```

```
fn1 called
Result is: true
```

参照

その他の技術情報

J# の演算子

~ 演算子 (Visual J#)

~ 演算子では、オペランドのビットごとの補数演算が実行されます。

```
~ expr
```

パラメータ

expr

プリミティブ整数型の式。

使用例

```
// operator_bitwise_compl.jsl

class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("~" + 8 + " = " + ~8);
        System.out.println("~" + -8 + " = " + ~-8);
    }
}
```

出力

```
~8 = -9
~-8 = 7
```

参照

その他の技術情報

J# の演算子

> 演算子 (Visual J#)

すべての数値型では、"より大きい" 関係演算子 (>) が定義されています。この演算子では、最初のオペランドが 2 番目のオペランドより大きい場合に **true** が返され、それ以外の場合は **false** が返されます。

```
expr1 > expr2
```

パラメータ

expr1

プリミティブ数値型の式。

expr2

プリミティブ数値型の式。

解説

オペランドの上位変換された型が **int** または **long** のときは、符号付き整数の比較が実行され、この上位変換された型が **float** または **double** のときは、浮動小数点の比較が実行されます。

使用例

```
// operator_greater_than.jsl

class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(1.1 > 1);
        System.out.println(1.1 > 1.1);
    }
}
```

出力

```
true
false
```

参照

その他の技術情報

J# の演算子

== 演算子 (Visual J#)

プリミティブ数値型、列挙型、値型、および **boolean** 型の場合、等値演算子 (==) は、オペランドの値が等しいときは **true** を返し、それ以外は **false** を返します。java.lang.String などの参照型の場合、== は、たとえ同じ値を格納していても、2 つのオペランドが異なるオブジェクトを参照しているときは **false** を返します。ただし、文字列定数の場合、== は文字列の値を比較し、2 つの文字列が同じであるときは **true** を返します。

```
expr1 == expr2
```

パラメータ

expr1

プリミティブ数値型、列挙型、値型、**boolean** 型、または参照型の式。

expr2

プリミティブ数値型、列挙型、値型、**boolean** 型、または参照型の式。

解説

オペランドの上位変換された型が **int** または **long** である場合は整数等価テストが実行され、上位変換された型が **float** または **double** の場合は浮動小数点等価テストが実行されます。

等値演算子のオペランドがどちらも **boolean** 型である場合、その演算のブール値 (**boolean**) は等価です。

等値演算子のオペランドがどちらも参照型か **null** 型である場合、その演算のオブジェクトは等価です。

等値演算子のオペランドがどちらも値型である場合、その演算の値は等価です。

使用例

```
// operator_equality.jsl

class MyClass
{
    public static void main(String[] args)
    {
    // Numeric equality:
    System.out.println((2 + 2) == 4);   // true

    // Reference equality: different Objects, same value.
    String s = new String("1");
    String t = new String("1");
    System.out.println(s == t);  // false

    // Define some Strings:
    String a = "hello";
    String b = String.Copy(a);
    String c = "hello";

    // Compare string values for a constant values.
    // true
    System.out.println(a == c);

    // Compare String objects;
    // a is a constant but b is an object:
    System.out.println(a == b);   // false
   }
}
```

出力

```
true
false
true
false
```

参照

関連項目

!= 演算子 (Visual J#)

その他の技術情報

J# の演算子

< 演算子 (Visual J#)

すべてのプリミティブ数値型では、"より小さい" 関係演算子 (<) が定義されています。この演算子では、最初のオペランドが 2 番目のオペランドより小さい場合に **true** が返され、それ以外の場合は **false** が返されます。

```
expr1 < expr2
```

パラメータ

expr1

プリミティブ数値型の式。

expr2

プリミティブ数値型の式。

解説

オペランドの上位変換された型が **int** または **long** のときは、符号付き整数の比較が実行され、この上位変換された型が **float** または **double** のときは、浮動小数点の比較が実行されます。

使用例

```
// operator_less_than.jsl

class Test
{
    public static void main(String[] args)
    {
        System.out.println(1 < 1.1);
        System.out.println(1.1 < 1.1);
    }
}
```

出力

```
true
false
```

参照

その他の技術情報

J# の演算子

| 演算子 (Visual J#)

二項 | 演算子は、整数プリミティブ型、列挙型、および **boolean** 型に対して組み込まれています。整数プリミティブ型と列挙型の場合、| ではオペランドのビットごとの OR が計算されます。**boolean** オペランドの場合は、オペランドの論理 OR が計算されます。つまり、両方のオペランドが **false** の場合だけ結果が **false** になります。

```
expr1 | expr2
```

パラメータ

expr1

プリミティブ整数型、列挙型、または **boolean** 型の式。

expr2

プリミティブ整数型、列挙型、または **boolean** 型の式。

使用例

```
// operator_OR.jsl

class MyClass
{
    public static void main(String[] args)
    {
        // Logical or:
        System.out.println(true | false);
        // Logical or:
        System.out.println(false | false);
        // Bitwise or:
        System.out.println(Integer.toHexString(0xf8 | 0x3f));
    }
}
```

出力

```
true
false
ff
```

参照

その他の技術情報

J# の演算子

+ 演算子 (Visual J#)

+ 演算子は、単項演算子または二項演算子として機能します。

```
+ expr
expr1 + expr2
```

パラメータ

expr

プリミティブ数値型または **enum** 型の式。

expr1

プリミティブ数値型または **String** 型の式。

expr2

プリミティブ数値型または **String** 型の式。

解説

単項 + 演算子は、すべての数値型および列挙型に対してあらかじめ定義されています。数値型または列挙型での単項 + 演算の結果は、単にオペランドの値になります。

二項 + 演算子は、数値型と文字列型に対してあらかじめ定義されています。数値型の場合、+ は 2 つのオペランドの合計を計算します。式の上位変換された型が **int** または **long** のときは、整数算術演算が実行され、この上位変換された型が **float** または **double** のときは、浮動小数点算術演算が実行されます。

オペランドの片方または両方が **String** 型の場合は、+ でオペランドの文字列表現が連結されます。

使用例

```
// operator_plus.jsl

class MyClass
{
    public static void main(String[] args)
    {
        // Unary plus:
        System.out.println(+5);
       // Addition:
        System.out.println(5 + 5);
        // Addition:
        System.out.println(5 + .5);
        // String concatenation.
        // Note automatic conversion from double to string:
        System.out.println("5" + "5");
    }
}
```

出力

```
5
10
5.5
55
```

参照

その他の技術情報

J# の演算子

>> 演算子 (Visual J#)

符号付き右シフト演算子 (**>>**) では、2 番目のオペランドで指定されたビット数だけ最初のオペランドが右にシフトされます。

```
expr >> count
```

パラメータ

Expr

プリミティブ整数型の式 (シフトされる値)。

Count

プリミティブ整数型の式 (シフト数)。

解説

expr の上位変換された型が **int** (32 ビット値) の場合、シフト数は *count* の下位 5 ビット (*count* & 0x1f) で指定されます。

expr の上位変換された型が **long** (64 ビット値) の場合、シフト数は *count* の下位 6 ビット (*count* & 0x3f) で指定されます。

符号付き右シフトは、符号拡張を使用する算術シフトです。

使用例

```
// operator_right_shift.jsl

class MyClass
{
    public static void main(String[] args)
    {
        int i = -1000;
        System.out.println(i >> 3);
    }
}
```

出力

```
-125
```

参照

関連項目

<< 演算子 (Visual J#)

その他の技術情報

J# の演算子

<< 演算子 (Visual J#)

左シフト演算子 (<<) では、2 番目のオペランドで指定されたビット数だけ最初のオペランドが左にシフトされます。

```
expr << count
```

パラメータ

expr

プリミティブ整数型の式 (シフトされる値)。

count

プリミティブ整数型の式 (シフト数)。

解説

expr の上位変換された型が **int** のときは、`count (count & 0x1f)` の下位 5 ビットがシフト カウントになります。

expr の上位変換された型が **long** のときは、`count (count & 0x3f)` の下位 6 ビットがシフト カウントになります。

expr の上位ビットは破棄され、シフトの結果空になる下位ビットには 0 が入ります。シフト演算では、オーバーフローは発生しません。

使用例

```
// operator_left_shift.jsl

class MyClass
{
    public static void main(String[] args)
    {
        int i = 1;
        long lg = 1;

        // Shift and convert to hexadecimal:
        System.out.println(Integer.toHexString(i << 1));
        System.out.println(Integer.toHexString(i << 33));
        System.out.println(Integer.toHexString((int)lg << 33));
    }
}
```

出力

```
2
2
2
```

1 と 33 では下位 5 ビットが同じであるため、`i<<1` と `i<<33` の結果は同じになります。

参照

関連項目

>> 演算子 (Visual J#)

その他の技術情報

J# の演算子

* 演算子 (Visual J#)

乗算演算子 (*) は、オペランドの積を計算します。

```
expr1 * expr2
```

パラメータ

expr1

プリミティブ数値型の式。

expr2

プリミティブ数値型の式。

解説

すべての数値型には定義済みの乗算演算子があります。上位変換された型が **int** または **long** のときは、整数算術演算が実行され、この上位変換された型が **float** または **double** のときは、浮動小数点算術演算が実行されます。

使用例

```
// operator_mult.jsl

class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(5 * 2);
        System.out.println(-.5 * .2);
    }
}
```

出力

```
10
-0.1
```

参照

その他の技術情報

J# の演算子

^ 演算子 (Visual J#)

二項 ^ 演算子は、プリミティブ型、列挙型、および **boolean** に対して組み込まれています。プリミティブ型と列挙型の場合、^ ではオペランドのビットごとの排他的 OR が計算されます。**boolean** オペランドの場合は、オペランドの排他的論理和が計算されます。つまり、片方のオペランドが **true** の場合だけ結果が **true** になります。

```
expr1 ^ expr2
```

パラメータ

expr1

プリミティブ整数型、列挙型、または **boolean** 型の式。

expr2

プリミティブ整数型、列挙型、または **boolean** 型の式。

使用例

```
// operator_bitwise_OR.jsl

class MyClass
{
    public static void main(String[] args)
    {
        // Logical exclusive-OR:
        System.out.println(true ^ false);
       // Logical exclusive-OR:
        System.out.println(false ^ false);
       // Bitwise exclusive-OR (converted to hexadecimal):
        System.out.println(Integer.toHexString(0xf8 ^ 0x3f));
    }
}
```

出力

```
true
false
c7
```

参照

その他の技術情報

J# の演算子

& 演算子 (Visual J#)

& 演算子は、論理演算子またはビットごとの AND 演算子として機能します。

```
expr1 & expr2
```

パラメータ

expr1

プリミティブ数値型、列挙型、または **boolean** 型の式。

expr2

プリミティブ数値型、列挙型、または **boolean** 型の式。

解説

二項 & 演算子は、プリミティブ数値型、列挙型、および **boolean** 型に対して定義されています。プリミティブ型と列挙型の場合、& ではオペランドのビットごとの AND が計算されます。**boolean** オペランドの場合は、オペランドの論理 AND が計算されます。つまり、両方のオペランドが **true** の場合だけ結果が **true** になります。

& 演算子は、最初の演算子の値に関係なく両方の演算子を評価します。以下にサンプルを示します。

```
int i = 0;

if (false & ++i == 1)

System.out.println("You won't get this output (the expression is false), but i == 1");
```

使用例

```
// operator_ampersand.jsl

class MyClass
{
    public static void main(String[] args)
    {
        // Logical and:
        System.out.println(true & false);
        // Logical and:
        System.out.println(true & true);
        // Bitwise and (converted to hexadecimal):
        System.out.println(Integer.toHexString(0xf8 & 0x3f));
    }
}
```

出力

```
false
true
38
```

参照

その他の技術情報

J# の演算子

<= 演算子 (Visual J#)

すべてのプリミティブ数値型では、"より小さいか等しい" 関係演算子 (<=) が定義されています。この演算子では、最初のオペランドが 2 番目のオペランドより小さいか等しい場合に **true** が返され、それ以外の場合は **false** が返されます。

```
expr1 <= expr2
```

パラメータ

expr1

プリミティブ数値型の式。

expr2

プリミティブ数値型の式。

解説

オペランドの上位変換された型が **int** または **long** のときは、符号付き整数の比較が実行され、この上位変換された型が **float** または **double** のときは、浮動小数点の比較が実行されます。

使用例

```
// operator_less_than_or_equal.jsl

class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(1 <= 1.1);
        System.out.println(1.1 <= 1.1);
    }
}
```

出力

```
true
true
```

参照

その他の技術情報

J# の演算子

/ 演算子 (Visual J#)

除算演算子 (/) では、最初のオペランドが 2 番目のオペランドで除算されます。すべてのプリミティブ数値型には定義済みの除算演算子があります。

```
expr1 / expr2
```

パラメータ

expr1

プリミティブ数値型の式。

expr2

プリミティブ数値型の式。

解説

使用例

```
// operator_division.jsl

class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(-5/2);
        System.out.println(-5.0/2);
    }
}
```

出力

```
-2
-2.5
```

参照

その他の技術情報

J# の演算子

++ 演算子 (Visual J#)

インクリメント演算子 (++) では、オペランドが 1 ずつインクリメントされます。インクリメント演算子は、次のようにオペランドの前または後に指定できます。

```
++ var
var ++
```

パラメータ

Var

プリミティブ数値型の式。

解説

最初の形式は、前置インクリメント演算です。この演算の結果は、インクリメントが行われた後のオペランドの値になります。

2 番目の形式は、後置インクリメント演算です。この演算の結果は、インクリメントが行われる前のオペランドの値になります。

final として宣言されている変数は、インクリメント演算子のオペランドとしては使用できません。

使用例

```
// operator_increment.jsl

class MyClass
{
    public static void main(String[] args)
    {
        double x;
        x = 1.5;
        System.out.println(++x);
        x = 1.5;
        System.out.println(x++);
        System.out.println(x);
    }
}
```

出力

```
2.5
1.5
2.5
```

参照

その他の技術情報

J# の演算子

!= 演算子 (Visual J#)

非等値演算子 (!=) では、オペランドが等しい場合に **false** が返され、それ以外の場合は **true** が返されます。

```
expr1 != expr2
```

パラメータ

expr1

式。

expr2

式。

解説

プリミティブ型、列挙型、および値型の場合、非等値演算子 (!=) ではオペランドの値が異なる場合に **true** が返され、それ以外の場合は **false** が返されます。参照型 (**String** を含む) または **null** 型の場合、!= では 2 つのオペランドが異なるオブジェクトを参照する場合に **true** が返されます。ただし、**String** 型については、変数にリテラル文字列を直接割り当てることによってオブジェクトを作成できます。この場合、!= は文字列の値を比較し、これらが正確に一致する場合は **false** を返します。

使用例

```
// operator_inequality.jsl

class MyClass
{
    public static void main(String[] args)
    {
        // Numeric inequality:
        System.out.println(4 != (2 + 2));   // false

        // Reference equality: different objects, same value:
        String s = new String("1");
        String t = new String("1");
        // Compare the two objects:
        // true
        System.out.println(s != t);

        // String equality: different variables, same value:
        String a = "hello";
        String b = "hello";
        // Compare the two strings:
        System.out.println(a != b);   // false
    }
}
```

出力

```
false
true
false
```

参照

関連項目

== 演算子 (Visual J#)

その他の技術情報

J# の演算子

! 演算子 (Visual J#)

論理補数演算子 (!) は、オペランドの補数演算を実行する単項演算子です。**boolean** に対して定義され、オペランドが **false** の場合のみ **true** を返します。

```
! expr
```

パラメータ

expr

boolean 型の式。

使用例

```
// operator_negation.jsl

class Test
{
    public static void main(String[] args)
{
        System.out.println(!true);
        System.out.println(!false);
    }
}
```

出力

```
false
true
```

参照

その他の技術情報

J# の演算子

%= 演算子 (Visual J#)

剰余代入演算子です。

```
lhs %= expr
```

パラメータ

lhs

格納場所。

expr

式。

解説

次のような %= 代入演算子を使用する式があるとします。

```
x %= y
```

上の式は、下の式と同等です。

```
x = x % y
```

ただし、x が評価されるのは 1 回だけです。% 演算子 (Visual J#) は、除算後の剰余を計算するために数値型に対して組み込まれています。

使用例

```
Code
// js_operator_modulus_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 5;
        a %= 3;
        System.out.println(a);
    }
}
```

出力

```
2
```

参照

関連項目

% 演算子 (Visual J#)

その他の技術情報

J# の演算子

&= 演算子 (Visual J#)

AND 代入演算子です。

```
lhs &= expr
```

パラメータ

lhs

格納場所。

expr

任意の式を指定します。

解説

次のような &= 代入演算子を使用する式があるとします。

```
x &= y
```

上の式は、下の式と同等です。

```
x = x & y
```

ただし、x が評価されるのは 1 回だけです。& 演算子 (Visual J#) では、整数のオペランドではビットごとの AND 演算、**boolean** オペランドでは論理 AND が実行されます。

使用例

```
// js_operator_and_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 0x0c;
        a &= 0x06;
        System.out.println("0x" + Integer.toHexString(a));
        boolean b = true;
        b &= false;
        System.out.println(b);
    }
}
```

出力

```
0x4
false
```

参照

関連項目

& 演算子 (Visual J#)

その他の技術情報

J# の演算子

() 演算子 (Visual J#)

かっこは、式の演算順序を指定するだけでなく、キャスト (型変換) を指定するためにも使用します。

```
( type ) expr
```

パラメータ

type

expr が変換される型の名前。

expr

式。

解説

キャストでは、*expr* の型から *type* に値が明示的に変換されます。このような型変換が定義されていない場合、キャストは失敗します。

使用例

次のプログラムは、**double** を **int** にキャストします。このプログラムは、キャストしないとコンパイルできません。

```
// js_operator_parentheses.jsl

class Test
{
    public static void main(String[] args)
    {
        double x = 1234.7;
        int a;
        // cast double to int:
        a = (int)x;
        System.out.println(a);
    }
}
```

出力

```
1234
```

参照

その他の技術情報

J# の演算子

*= 演算子 (Visual J#)

乗算代入演算子です。

```
lhs *= expr
```

パラメータ

lhs

格納場所。

expr

式。

解説

次のような *= 代入演算子を使用する式があるとします。

```
x *= y
```

これは次と同じです。

```
x = x * y
```

ただし、x が評価されるのは 1 回だけです。* 演算子 (Visual J#) は、乗算のために数値型に対して組み込まれています。

使用例

```
// js_operator_multiplication_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 5;
        a *= 6;
        System.out.println(a);
    }
}
```

出力

```
30
```

参照

関連項目

* 演算子 (Visual J#)

その他の技術情報

J# の演算子

. 演算子 (Visual J#)

メンバ アクセス演算子。

ドット演算子 (.) は、メンバ アクセスに使用します。ドット演算子は、メソッドやクラス、またはパッケージ内のその他のメンバを指定します。またはサブパッケージを指定します。ドット演算子は、このサンプルにあるとおり、J# クラス ライブラリ内の特定のメソッドにアクセスするために広範囲に使用されます。

```
// class System in package java.lang.
java.lang.System.out.println("hello");
```

```
name1 . name2
```

パラメータ

name1

パッケージまたは型。

name2

パッケージ、型、メソッド、またはフィールド。

解説

たとえば、次のクラスを考えます。

```
class Simple
{
    public int a;
    public void b()
    {
    }
}
Simple s = new Simple();
```

変数 `s` には、`a` と `b` という 2 つのメンバがあります。それらのメンバにアクセスするためにドット演算子を使用します。

```
// Assign to field a:
s.a = 6;
// Invoke member function b:
s.b();
```

ドットは修飾名の記述にも使用します。修飾名とは、属しているパッケージやインターフェイスなどを示す名前のことです。

```
// class System in package java.lang.
java.lang.System.out.println("hello");
```

import ディレクティブを使用すると、名前の修飾を省略できます。

```
import java.lang.System;

java.lang.System.out.println("hello");
// same thing
System.out.println("hello");
```

ただし、識別子があいまいな場合は、修飾する必要があります。

```
import java.lang.System;
// A package containing another System class:
```

```
import OtherSystem;

// Must qualify System:
java.lang.System.out.println( "hello" );
```

参照

その他の技術情報

J# の演算子

/= 演算子 (Visual J#)

除算代入演算子です。

```
lhs /= expr
```

パラメータ

lhs

格納場所。

expr

式。

解説

次のような /= 代入演算子を使用する式があるとします。

```
x /= y
```

上の式は、下の式と同等です。

```
x = x / y
```

ただし、x が評価されるのは 1 回だけです。/ 演算子 (Visual J#) は、除算のために数値型に対して組み込まれています。

使用例

```
// js_operator_division_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 5;
        a /= 6;
        System.out.println(a);
        double b = 5;
        b /= 6;
        System.out.println(b);
    }
}
```

出力

```
0
0.83333333333333337
```

参照

関連項目

/ 演算子 (Visual J#)

その他の技術情報

J# の演算子

?: 演算子 (Visual J#)

条件演算子 (?:) では、ブール式の値に応じて 2 つの値のいずれかが返されます。条件演算子は、次の形式の式で使用します。

```
cond-expr ? expr1 : expr2
```

パラメータ

cond-expr

boolean 型の式。

expr1

任意の式を指定します。

expr2

任意の式を指定します。

解説

cond-expr が **true** のときは、*expr1* が評価され、その結果となります。*cond-expr* が **false** のときは、*expr2* が評価され、その結果となります。評価されるのは、*expr1* と *expr2* のいずれか一方だけです。

if-else の構造が必要になる計算は、条件演算子を使用すると、簡潔で明快に表現できます。たとえば、sine 関数の計算で 0 による除算を避けるには、次のいずれかを記述します。

```
if (x != 0.0)
{
    s = Math.sin(x)/x;
}
else
{
    s = 1.0;
}
```

条件演算子を使用すると、次のように記述できます。

```
s = x != 0.0 ? Math.sin(x)/x : 1.0;
```

条件演算子は結合規則が次のように右から左です。

```
a ? b : c ? d : e
```

この式は、次のように評価されます。

```
a ? b : (c ? d : e)
```

次のようには指定しないでください。

```
(a ? b : c) ? d : e
```

条件演算子は、オーバーロードできません。

使用例

```
// js_operator_conditional.jsl

class Test
{
    public static double sin(double x)
```

```
    {
        return x != 0.0 ? Math.sin(x)/x : 1.0;
    }

    public static void main(String[] args)
    {
        System.out.println(sin(0.2));
        System.out.println(sin(0.1));
        System.out.println(sin(0.0));
    }
}
```

出力

```
0.99334665397530608
0.99833416646828155
1.0
```

参照

関連項目

if (Visual J#)

その他の技術情報

J# の演算子

[] 演算子 (Visual J#)

角かっこ ([]) は、配列の宣言、および配列内のインデックスの指定に使用されます。

```
type []
array [ indexexpr ]
```

パラメータ

type

型。

array

配列。

indexexpr

インデックス式。

解説

配列型は、型名の後に [] が続きます。

```
// fib is of type int[], "array of int"
int[] fib;
// Create a 100-element int array:
fib = new int[100];
```

配列の要素にアクセスするには、目的の要素の添字を角かっこで囲みます。

```
fib[0] = fib[1] = 1;
for (int i=2; i<100; ++i)
{
    fib[i] = fib[i-1] + fib[i-2];
}
```

配列の添字が範囲外の場合は、例外がスローされます。

参照

その他の技術情報

J# の演算子

^= 演算子 (Visual J#)

排他的 OR 代入演算子です。

```
lhs ^= expr
```

パラメータ

lhs

格納場所。

expr

式。

解説

次のような形式の式があるとします。

```
x ^= y
```

この式は、次のように評価されます。

```
x = x ^ y
```

ただし、x が評価されるのは 1 回だけです。^ 演算子 (Visual J#) では、整数のオペランドではビットごとの排他的 OR 演算、**boolean** オペランドでは排他的論理和が実行されます。

使用例

```
// js_operator_xor_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 0x0c;
        a ^= 0x06;
        System.out.println("0x" + Integer.toHexString(a));
        boolean b = true;
        b ^= false;
        System.out.println(b);
    }
}
```

出力

```
0xa
true
```

参照

関連項目

^ 演算子 (Visual J#)

その他の技術情報

J# の演算子

|= 演算子 (Visual J#)

OR 代入演算子です。

```
lhs |= expr
```

パラメータ

lhs

格納場所。

expr

任意の式を指定します。

解説

次のような |= 代入演算子を使用する式があるとします。

```
x |= y
```

上の式は、下の式と同等です。

```
x = x | y
```

ただし、x が評価されるのは 1 回だけです。| 演算子 (Visual J#) では、整数のオペランドではビットごとの OR 演算、**boolean** オペランドでは論理 OR が実行されます。

使用例

```
// js_operator_or_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 0x0c;
        a |= 0x06;
        System.out.println("0x" + Integer.toHexString(a));
        boolean b = true;
        b |= false;
        System.out.println(b);
    }
}
```

出力

```
0xe
true
```

参照

関連項目

| 演算子 (Visual J#)

その他の技術情報

J# の演算子

+= 演算子 (Visual J#)

加算代入演算子です。

```
lhs += expr
```

パラメータ

lhs

格納場所。

expr

任意の式を指定します。

解説

次のような += 代入演算子を使用する式があるとします。

```
x += y
```

上の式は、下の式と同等です。

```
x = x + y
```

ただし、x が評価されるのは 1 回だけです。+ 演算子 (Visual J#) の意味は、x および y の型に依存します。たとえば、数値オペランドの場合は加算、文字列オペランドの場合は連結になります。

使用例

```
// js_operator_addition_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 5;
        a += 6;
        System.out.println(a);
        String s = "Micro";
        s += "soft";
        System.out.println(s);
    }
}
```

出力

```
11
Microsoft
```

参照

関連項目

+ 演算子 (Visual J#)

その他の技術情報

J# の演算子

<<= 演算子 (Visual J#)

左シフト代入演算子です。

```
lhs <<= expr
```

パラメータ

lhs

格納場所。

expr

任意の式を指定します。

解説

次のような形式の式があるとします。

```
x <<= y
```

この式は、次のように評価されます。

```
x = x << y
```

ただし、x が評価されるのは 1 回だけです。<< 演算子 (Visual J#) では、y で指定された分だけ x が左にシフトされます。

使用例

```
// js_operator_left_shift_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 1000;
        a <<= 4;
        System.out.println(a);
    }
}
```

出力

```
16000
```

参照

関連項目

<< 演算子 (Visual J#)

その他の技術情報

J# の演算子

= 演算子 (Visual J#)

代入演算子 (=) では、右辺のオペランドの値が左辺のオペランドで示された格納場所に格納され、その値が結果として返されます。両側のオペランドは、同じ型である必要があります。同じ型でない場合、右辺のオペランドは、左辺のオペランドの型に暗黙に変換できる必要があります。

```
lhs = expr
```

パラメータ

lhs

格納場所。

expr

式。

使用例

```
// js_operator_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        double x;
        int i;
        // int to int assignment.
        i = 5;
        // Implicit conversion from int to double.
        x = i;
        // Needs cast:
        i = (int)x;
        System.out.println("i is " + i + ", x is " + x);
        Integer objInt = new Integer(i);
        System.out.println("boxed value = " + objInt +
            ", class is " + objInt.getClass());
        i = objInt.intValue();
        System.out.println("unboxed: " + i);
    }
}
```

出力

```
i is 5, x is 5.0
boxed value = 5, class is class java.lang.Integer
unboxed: 5
```

参照

その他の技術情報

J# の演算子

-= 演算子 (Visual J#)

減算代入演算子です。

```
lhs -= expr
```

パラメータ

lhs

格納場所。

expr

式。

解説

次のような -= 代入演算子を使用する式があるとします。

```
x -= y
```

これは次と同じです。

```
x = x - y
```

ただし、x が評価されるのは 1 回だけです。

使用例

```
// js_operator_subtraction_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 5;
        a -= 6;
        System.out.println(a);
    }
}
```

出力

```
-1
```

参照

関連項目

- 演算子 (Visual J#)

その他の技術情報

J# の演算子

>>= 演算子 (Visual J#)

右シフト代入演算子です。

```
lhs >>= expr
```

パラメータ

lhs

格納場所。

expr

式。

解説

次のような形式の式があるとします。

```
x >>= y
```

この式は、次のように評価されます。

```
x = x >> y
```

ただし、x が評価されるのは 1 回だけです。>> 演算子 (Visual J#) では、y で指定された分だけ x が右にシフトされます。

使用例

```
// js_operator_right_shift_assignment.jsl

class Test
{
    public static void main(String[] args)
    {
        int a = 1000;
        a >>= 4;
        System.out.println(a);
    }
}
```

出力

```
62
```

参照

関連項目

>> 演算子 (Visual J#)

その他の技術情報

J# の演算子

>>> 演算子 (Visual J#)

符号なし右シフト演算子 (>>) では、2 番目のオペランドで指定されたビット数だけ最初のオペランドが右にシフトされます。

```
expr >>> count
```

パラメータ

Expr

プリミティブ整数型の式 (シフトされる値)。

Count

プリミティブ整数型の式 (シフト数)。

解説

expr の上位変換された型が **int** (32 ビット値) の場合、シフト数は *count* の下位 5 ビット (*count* & 0x1f) で指定されます。

expr の上位変換された型が **long** (64 ビット値) の場合、シフト数は *count* の下位 6 ビット (*count* & 0x3f) で指定されます。

符号なし右シフトは、ゼロ拡張を使用する算術シフトです。

使用例

```
// operator_right_shift.jsl

class MyClass
{
    public static void main(String[] args)
    {
        int i = -1000;
        System.out.println(i >>> 3);
    }
}
```

出力

```
536870787
```

参照

関連項目

>> 演算子 (Visual J#)

<< 演算子 (Visual J#)

その他の技術情報

J# の演算子

Visual J# コンパイラ

コンパイラは、実行可能ファイル (.exe)、ダイナミック リンク ライブラリ ファイル (.dll)、およびモジュール ファイル (.netmodule) を生成するツールです。

各コンパイラ オプションには、それぞれ **-option** と **/option** という 2 つの形式があります。ドキュメントでは **/option** の形式だけを示しています。**-option** の形式を使用する場合も内容はまったく同じなので、適宜置き換えてお読みください。

このセクションの内容

コマンド ラインからのビルド

コマンド ラインでの Visual J# アプリケーションのビルドについて説明します。

Visual J# コンパイラ オプション一覧 (カテゴリ別)

コンパイラ オプションのカテゴリ別の一覧です。

Visual J# コンパイラ オプション一覧 (アルファベット順)

コンパイラ オプションのアルファベット順の一覧です。

Visual J# コンパイラ オプションに対応する Visual J++ コンパイラ オプション

Visual J++® 6.0 のコンパイラ オプションと Visual J# のコンパイラ オプションとの対応について説明します。

コンパイラ エラー VJS0001 から VJS9999

Visual J# コンパイラがスローするエラーについて説明します。

関連するセクション

Visual J# プロジェクトのプロパティの設定

プロジェクトのコンパイル方法、ビルド方法、およびデバッグ方法を制御するプロパティの一般的な設定手順の一覧です。カスタム ビルド ステップに関する情報も含まれています。

既定のビルドとカスタム ビルド

ビルドの種類と構成について説明します。

ビルドの準備と管理

Visual Studio® .NET 開発環境でのビルド手順について説明します。

コマンド ラインからのビルド (Visual J#)

Visual J# コンパイラは、コマンド ラインでコンパイラの実行可能ファイルの名前 (vjc.exe) を入力することにより、コマンド ラインからも起動できます。コンピュータの任意のサブディレクトリから vjc.exe を起動するには、パスの調整が必要になる場合があります。

Visual J# のサンプルをコマンド ラインからビルドおよび実行する場合は、<%Program Files%>\Microsoft Visual Studio 8\Common7\Tools ディレクトリにある vsVars.bat を実行してください。代わりに、Visual Studio コマンド プロンプトを使用することもできます。このコマンド プロンプトを使用するには、[スタート] メニューの [すべてのプログラム] をポイントし、[Microsoft Visual Studio] をポイントして、[Visual Studio Tools] をクリックします。

/debug などのブール型のオプションでは、**+** と **–** を使ってオプションを有効または無効にできます。たとえば、デバッグ情報の生成を有効にするには、**/debug+** または **/debug** を指定します。

コマンド ラインでは、オプションおよびソース コード ファイル名が特定の順序で指定されていなくてもかまいません。

メモ :

既定では、これらの .NET Framework ライブラリは、vjslib.dll、vjscor.dll、vjslibcw.dll、mscorlib.dll、System.dll によって参照されます。

コマンド ライン構文の規則

Visual J# コンパイラのコードは、オペレーティング システム (OS) のコマンド ラインで指定された引数を次の規則に従って解釈します。

- 引数は、空白 (スペースまたはタブ) で区切ります。
- キャレット (^) は、エスケープ文字やデリミタとしては認識されません。キャレットは、OS のコマンド ライン パーサーによって完全に処理されてからプログラムの `argv` 配列に渡されます。
- 二重引用符で囲まれた文字列 ("*string*") は、空白を含む場合でも、単一の引数と見なされます。二重引用符で囲んだ文字列を引数に埋め込むこともできます。
- 円記号を前に付けた二重引用符 (\") は、リテラル二重引用符文字 (") として解釈されます。
- 二重引用符の直前にある円記号以外は、円記号 (\) として解釈されます。
- 二重引用符の直前に円記号が偶数個 (0 個は含まない) あると、円記号のペアごとに 1 個の円記号が `argv` 配列に格納されます。この場合、二重引用符は文字列のデリミタとして解釈されます。
- 二重引用符の直前に円記号が奇数個 (3 個以上) あると、円記号のペアごとに 1 個の円記号が `argv` 配列に格納されます。最後の円記号によって二重引用符がエスケープ シーケンスになるため、二重引用符文字 (") がそのまま `argv` に格納されます。

コマンド ラインの例

- File.jsl をコンパイルして File.exe を作成するには、次のコードを使用します。

```
vjc File.jsl
```

- File.jsl をコンパイルして File.dll を作成するには、次のコードを使用します。

```
vjc /target:library File.jsl
```

- File.jsl をコンパイルして My.exe を作成するには、次のコードを使用します。

```
vjc /out:My.exe File.jsl
```

- **DEBUG** シンボルを定義して、現在のディレクトリにあるすべての .jsl ファイルをコンパイルします。File2.exe が出力されます。

```
vjc /define:DEBUG /out:File2.exe *.jsl
```

- 現在のディレクトリにあるすべての Visual J# ファイルをコンパイルして、デバッグ バージョンの File2.dll を作成します。ここでは、**/target** オプションの省略形が使用されています。ロゴは表示されません。

```
vjc /t:l /out:File2.dll /nologo /debug *.jsl
```

- 現在のディレクトリと各サブディレクトリにあるすべての .jsl ファイルをコンパイルし、MyLibrary.dll を生成します。

```
vjc /target:library /recurse:*.jsl /out:MyLibrary.dll
```

- MyLibrary.dll からクラスをインポートし、main.jsl をコンパイルします。

```
vjc /reference:MyLibrary.dll main.jsl
```

参照

その他の技術情報

Visual J# コンパイラ
Visual J# コンパイラ オプション一覧 (アルファベット順)
Visual J# コンパイラ オプション一覧 (カテゴリ別)

Visual J# コンパイラ オプション一覧 (カテゴリ別)

次のコンパイラ オプションは、カテゴリ別に並んでいます。アルファベット順の一覧については、「Visual J# コンパイラ オプション一覧 (アルファベット順)」を参照してください。

最適化

オプション	目的
/optimize	最適化を有効または無効にします。

出力ファイル

オプション	目的
/out	出力ファイルを指定します。
/target	次のいずれかのオプションを使用して、出力ファイルの形式を指定します。 /target:exe /target:library /target:module /target:winexe

.NET Framework アセンブリ

オプション	目的
/delaysignn	暗号化キーを追加します。ただし、アセンブリへの署名は行いません。
/keycontainer	暗号化キー コンテナの名前を指定します。
/keyfile	暗号化キーの格納されたファイルを指定します。
/libpath	/reference を使って参照されるアセンブリの場所を指定します。
/reference	アセンブリを含むファイルからメタデータをインポートします。
/securescoping	パッケージ スコープを持つメンバのアクセスを、アセンブリの内部に制限します。

デバッグ/エラーのチェック

オプション	目的
/debug	デバッグ情報を生成します。
/nowarn	指定された警告を生成するコンパイラの機能を無効にします。
/warn	警告レベルを設定します。
/warnaserror	警告をエラーとして扱います。

プリプロセッサ

オプション	目的
/define	プリプロセッサ シンボルを定義します。

リソース

オプション	目的
/linkresource	.NET Framework リソースへのリンクを出力ファイルに作成します。
/resource	.NET Framework リソースを出力ファイルに埋め込みます。
/win32res	Win32 リソースを出力ファイルに挿入します。

その他

オプション	目的
@	応答ファイルを指定します。
/?	標準出力にコンパイラ オプションの一覧を表示します。
/baseaddress	DLL を読み込むベース アドレスを指定します。
/codepage	コンパイルですべてのソース コード ファイルに使用するコード ページを指定します。
/help	標準出力にコンパイラ オプションの一覧を表示します。
/jcpa	Java 言語/COM パッケージの関連付けを指定します。
/main	**main** メソッドの場所を指定します。
/nologo	コンパイラの著作権情報が表示されないようにします。
/recurse	コンパイルするソース ファイルをサブディレクトリで検索します。
/utf8output	UTF-8 エンコーディングを使用してコンパイラ出力を表示します。
/x	言語拡張機能を無効にします。

参照

関連項目

コマンド ラインからのビルド (Visual J#)

その他の技術情報

Visual J# コンパイラ

Visual J# コンパイラ オプション一覧 (アルファベット順)

Visual J# コンパイラ オプション一覧 (アルファベット順)

次のコンパイラ オプションは、アルファベット順に並んでいます。カテゴリ別の一覧については、「Visual J# コンパイラ オプション一覧 (カテゴリ別)」を参照してください。

オプション	目的
@	応答ファイルを指定します。
/?	標準出力にコンパイラ オプションの一覧を表示します。
/baseaddress	DLL を読み込むベース アドレスを指定します。
/codepage	コンパイルですべてのソース コード ファイルに使用するコード ページを指定します。
/debug	デバッグ情報を生成します。
/define	プリプロセッサ シンボルを定義します。
/delaysign	暗号化キーを追加します。ただし、アセンブリへの署名は行いません。
/help	標準出力にコンパイラ オプションの一覧を表示します。
/jcpa	Java 言語/COM パッケージの関連付けを指定します。
/keycontainer	暗号化キー コンテナの名前を指定します。
/keyfile	暗号化キーの格納されたファイルを指定します。
/libpath	/reference を使って参照されるアセンブリの場所を指定します。
/linkresource	.NET Framework リソースへのリンクを出力ファイルに作成します。
/main	**main** メソッドの場所を指定します。
/nologo	コンパイラの著作権情報が表示されないようにします。
/nowarn	指定された警告を生成するコンパイラの機能を無効にします。
/optimize	最適化を有効または無効にします。
/out	出力ファイルを指定します。
/recurse	コンパイルするソース ファイルをサブディレクトリで検索します。
/reference	アセンブリを含むファイルからメタデータをインポートします。
/resource	.NET Framework リソースを出力ファイルに埋め込みます。
/securescoping	パッケージ スコープを持つメンバのアクセスを、アセンブリの内部に制限します。
/target	/target:exe/target:library/target:module/target:winexe のいずれかのオプションを使用して出力ファイルの形式を指定します。

/utf8output	UTF-8 エンコーディングを使用してコンパイラ出力を表示します。
/warn	警告レベルを設定します。
/warnaserror	警告をエラーとして扱います。
/win32res	Win32 リソースを出力ファイルに挿入します。
/x	言語拡張機能を無効にします。

参照

関連項目

コマンド ラインからのビルド (Visual J#)

その他の技術情報

Visual J# コンパイラ

Visual J# コンパイラ オプション一覧 (カテゴリ別)

@ (応答ファイルの指定) (Visual J#)

コンパイラ オプションと、コンパイルするソース コード ファイルを含むファイルを指定します。これらのコンパイラ オプションとソース コード ファイルは、コマンド ラインで指定した場合と同じように、コンパイラによって処理されます。

```
@response_file
```

引数
response_file

コンパイラ オプションやコンパイルするソース コード ファイルの一覧を含むファイル。

解説

コンパイル時に複数の応答ファイルを指定するには、複数の応答ファイル オプションを指定します。次に例を示します。

```
@file1.rsp @file2.rsp
```

応答ファイルでは、複数のコンパイラ オプションとソース コード ファイルを 1 行に記述できます。1 つのコンパイラ オプションは 1 行に指定する必要があり、複数行にわたって指定できません。応答ファイルには、シャープ記号 (#) で始まるコメントを記述できます。

応答ファイルでのコンパイラ オプションの指定方法は、コマンド ラインでのコンパイラ オプションの指定方法と同じです。詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

コンパイラでは、検出順にコマンド オプションを処理します。このため、コマンド ライン引数によって、応答ファイルで先に指定したオプションをオーバーライドできます。逆に、応答ファイルのオプションが、コマンド ラインや他の応答ファイルで先に指定したオプションをオーバーライドすることもあります。

vsprvs 開発環境でこのコンパイラ オプションを設定するには

- このコンパイラ オプションは、vsprvs では利用できません。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

使用例

サンプルの応答ファイルの一部を次に示します。

```
# build the first output file
/target:exe /out:MyExe.exe source1.jsl source2.jsl
```

参照
関連項目
[ビルドの詳細設定] ダイアログ ボックス (J#)
その他の技術情報
Visual J# コンパイラ

/baseaddress (DLL のベース アドレスの指定) (Visual J#)

DLL を読み込むベース アドレスを指定します。

```
/baseaddress:address
```

引数
address

DLL のベース アドレス。このアドレスは 16 進数として指定する必要があります。

解説

DLL の既定のベース アドレスは、.NET Framework 共通言語ランタイムによって設定されます。

このアドレスの下位ワードは丸められることに注意してください。たとえば、0x11110001 と指定すると、丸められて 0x11110000 になります。

ターゲットが DLL でない場合、このオプションは無視されます。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [詳細] をクリックします。
4. DLL のベース アドレス プロパティを変更します。

参照
関連項目
[ビルドの詳細設定] ダイアログ ボックス (J#)
その他の技術情報
Visual J# コンパイラ オプション一覧 (カテゴリ別)

/codepage (ソース コード ファイルのコード ページの指定) (Visual J#)

コンピュータの既定のコード ページ以外で作成されたソース コード ファイルをコンパイルする場合に、使用するコード ページを指定します。

```
/codepage:id
```

引数
id

コンパイル時にすべてのソース コード ファイルで使うコード ページの ID。

解説

/codepage は、コンパイル時にすべてのソース コード ファイルに適用されます。

ソース コードの作成時に使用されたコード ページがコンピュータで有効なコード ページと同じ場合、または UNICODE か UTF-8 の場合は、**/codepage** を使う必要はありません。

システムでサポートされているコード ページを探す方法については、「GetCPInfo」を参照してください。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [詳細] をクリックします。
4. [コード ページ] プロパティを変更します。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

参照
関連項目
[ビルドの詳細設定] ダイアログ ボックス (J#)
その他の技術情報
Visual J# コンパイラ

/debug (デバッグ情報の生成) (Visual J#)

コンパイラでデバッグ情報を生成し、出力ファイルを作成します。このオプションを使用してデバッグ ビルドを作成します。

```
/debug[+ | -]
/debug:{full | pdbonly}
```

引数

+ | -

+ を指定するか、または **/debug** だけを指定すると、コンパイラによってデバッグ情報が生成され、プログラム データベース (.pdb) ファイルにその情報が出力されます。- を指定すると、デバッグ情報は作成されません。**/debug** を指定しない場合は、- が有効になります。

full | pdbonly

コンパイラによって生成されるデバッグ情報の種類を指定します。**/debug:pdbonly** を指定しない場合、つまり full 引数を使用すると、実行中のプログラムにデバッガをアタッチできます。**pdbonly** を指定すると、プログラムがデバッガで開始されたときにはソース コードをデバッグできますが、実行中のプログラムをデバッガにアタッチしたときはアセンブラしか表示されません。

解説

/debug、**/debug+**、**/debug:full** のいずれも指定しなかった場合、プログラムの出力ファイルをデバッグすることはできません。

アプリケーションのデバッグ パフォーマンスを設定する方法の詳細については、「イメージのデバッグの簡略化」を参照してください。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [詳細] をクリックします。
4. [デバッグ情報] ボックスの一覧の [完全なデバッグ情報の生成] を選択します。

このコンパイラ オプションをコードから設定するには

- システム コール中のクラッシュのデバッグに必要なシンボルのインストール を参照してください。

使用例

デバッグ情報をファイル `app.pdb` に出力する例を次に示します。

```
vjc /debug /out:app.pdb test.jsl
```

参照

関連項目

[ビルドの詳細設定] ダイアログ ボックス (J#)

その他の技術情報

Visual J# コンパイラ

/define (プリプロセッサ定義) (Visual J#)

name をプログラム内のシンボルとして定義します。

```
/define:name[=value][;name2[=value]]
```

引数
name, name2

定義する 1 つ以上のシンボルの名前。

value

シンボルに割り当てる省略可能な値。

解説

このオプションには、ソース ファイルで **#define** プリプロセッサ ディレクティブを使用する場合と同じ効果があります。ソース ファイルの **#undef** ディレクティブがこの定義を削除するか、またはコンパイラがファイルの終端に達するまで、シンボルは削除されません。

このオプションで作成されるシンボルを、**#if**、**#else**、**#elif**、**#endif** と組み合わせて使用することにより、ソース ファイルを条件付きでコンパイルできます。

/d は **/define** の省略形です。

/define では、シンボル名をセミコロンまたはコンマで区切ることにより、複数のシンボルを定義できます。次に例を示します。

```
/define:DEBUG;TUESDAY
```

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [条件付きコンパイル シンボル] プロパティを変更します。

このコンパイラ オプションをコードから設定するには

- DefineConstants を参照してください。

使用例

```
// vjc_compiler_define.jsl
// compile with: /define:DEBUG
// or uncomment the next line
// #define DEBUG
public class Test
{
   public static void main(String args[])
   {
      #if (DEBUG)
         System.Console.WriteLine("DEBUG defined");
      #else
         System.Console.WriteLine("DEBUG not defined");
      #endif
   }
}
```

出力

```
DEBUG defined
```

参照

関連項目

[ビルドの詳細設定] ダイアログ ボックス (J#)

その他の技術情報

Visual J# コンパイラ オプション一覧 (カテゴリ別)

/delaysign (暗号化キーの遅延署名)

アセンブリの署名を格納するか、後で追加するかを指定します。

```
/delaysign[+ | -]
```

引数
+ | -

完全署名されたアセンブリを作成する場合は、**/delaysign-** を使用します。アセンブリに公開キーだけを含める場合は、**/delaysign+** を使います。既定値は **/delaysign-** です。

解説

/delaysign オプションは、/keyfile (厳密名キー ファイルの指定) (C# コンパイラ オプション) または /keycontainer (厳密名キー コンテナの指定) (C# コンパイラ オプション) と共に使用しなければ無効になります。

(オプションを省略するか、**/delaysign-** を使用して) 完全署名されたアセンブリを要求すると、コンパイラはマニフェスト (アセンブリ メタデータ) を含むファイルをハッシュし、そのハッシュに秘密キーで署名します。結果として得られるデジタル署名は、マニフェストを含むファイルに格納されます。アセンブリを遅延署名に設定すると、コンパイラは署名の計算も格納も行いませんが、後で署名を追加できるようにファイルに領域を確保します。

たとえば、**/delaysign+** を指定すると、テスト時にはアセンブリをグローバル キャッシュに格納できます。テスト後に、アセンブリ リンカ (Al.exe) ユーティリティを使用してアセンブリに秘密キーを追加することにより、そのアセンブリに完全署名できます。

詳細については、「厳密な名前付きアセンブリの作成と使用」を参照してください。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクトの [プロパティ] ページを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [署名] プロパティ ページをクリックします。
3. [遅延署名のみ] プロパティを変更します。

このコンパイラ オプションをプログラムで設定する方法については、「DelaySign」を参照してください。

参照
その他の技術情報
Visual J# コンパイラ

/help、/?(コンパイラのコマンド ラインのヘルプ) (Visual J#)

標準出力にコンパイラ オプションの一覧と各オプションの簡単な説明を出力します。

```
/help
/?
```

解説

コンパイル時にこのオプションを指定すると、出力ファイルは作成されず、コンパイルも実行されません。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

- このコンパイラ オプションは、Visual Studio では利用できません。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

参照

その他の技術情報

Visual J# コンパイラ

/jcpa (Java 言語/COM パッケージの関連付け)

アプリケーションがアクセスする 2 つ以上の COM DLL に同じ名前のクラスまたはインターフェイスが含まれている場合に使用します。この状況は、Visual J++ 6.0 の Java 言語/COM アプリケーションを Visual J# にアップグレードする場合に発生します。

```
/jcpa:package=namespace
/jcpa:@filename
```

引数

package

COM クラスまたはインターフェイスのパッケージの名前。

namespace

パッケージを対応付けるマネージ ラッパーの名前空間。

filename

package=namespace の 1 つ以上の関連付けが含まれるファイル。ファイル内で複数のペアを指定する場合は、各ペアを改行またはセミコロンで区切ります。

解説

JActiveX で生成した Java 言語/COM ラッパーのソースをコンパイルするときには、同じ名前の COM インターフェイスが 1 つ以上のパッケージに含まれていることがあります。この場合、同じ COM インターフェイスに対応するマネージ型が、タイプ ライブラリ インポータ ツール (Tlbimp.exe) によって生成された複数のアセンブリに存在することになります。このような状況では、vjc.exe は COM インターフェイスを適切なマネージ型に変換できないため、コンパイル時にエラーが発生します。この衝突を解決するには、**/jcpa** を使います。

詳細については、「概要 : Java 言語/COM コンポーネントのアップグレード」を参照してください。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [詳細] をクリックします。
4. [Java-COM パッケージの関連付け] プロパティを変更します。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

使用例

パッケージ `x` と名前空間 `y` を関連付けて `in.jsl` をコンパイルするには、次のコマンド ラインを使います。

```
vjc /jcpa:x=y in.jsl
```

参照

関連項目

[ビルドの詳細設定] ダイアログ ボックス (J#)

その他の技術情報

Visual J# コンパイラ

/keycontainer (暗号化キー コンテナの指定)

暗号化キー コンテナの名前を指定します。

```
/keycontainer:string
```

引数
string

厳密名キーのコンテナ名。

解説

/keycontainer オプションが指定された場合、コンパイラは、指定されたコンテナに格納された公開キーをアセンブリ マニフェストに追加し、最終的なアセンブリを秘密キーで署名することによって、共有可能なコンポーネントを生成します。キー ファイルを生成するには、コマンド ラインで「**sn -k file**」と入力します。**sn -i** は、キー ペアをコンテナに組み込みます。

/target:module (アセンブリに追加するモジュールの作成) (Visual J#) を使用してコンパイルすると、キー ファイル名はモジュールに保持され、さらに、そのモジュールが最終的にアセンブリにコンパイルされた場合、生成されたアセンブリに組み込まれます。

/keyfile (暗号化キーのファイル名の指定) を使用して、暗号に関する情報をコンパイラに渡すこともできます。公開キーのみアセンブリ マニフェストに追加しておき、アセンブリへの署名については、テストが終わるまで保留にする場合は、/delaysign (暗号化キーの遅延署名) を使用します。

詳細については、「厳密な名前付きアセンブリの作成と使用」を参照してください。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

- このコンパイラ オプションは、Visual Studio 開発環境では使用できません。

このコンパイラ オプションには、AssemblyKeyContainerName を使用してプログラムでアクセスできます。

参照
その他の技術情報
Visual J# コンパイラ

/keyfile (暗号化キーのファイル名の指定)

暗号化キーの格納されたファイル名を指定します。

```
/keyfile:file
```

解説
file

厳密名キーが格納されたファイルの名前。

解説

このオプションを使用した場合、コンパイラは、指定されたファイルから公開キーを取り出して、アセンブリ マニフェストに挿入します。**/delaysign+** が使用されない限り、最終的なアセンブリは秘密キーで署名されます。キー ファイルを生成するには、コマンド ラインで「**sn -k file**」と入力します。

/target:module を使用してコンパイルすると、キー ファイル名はモジュールに保持され、さらに、そのモジュールが最終的にアセンブリにコンパイルされた場合、生成されたアセンブリに組み込まれます。

/keycontainer (暗号化キー コンテナの指定) を使用して、暗号に関する情報をコンパイラに渡すこともできます。部分署名されたアセンブリを作成する場合は、/delaysign (暗号化キーの遅延署名) を使用します。

コマンド ライン オプションまたはカスタム属性によって、コンパイル時に **/keyfile** と **/keycontainer** の両方が同時に指定されると、コンパイラは先にキー コンテナを処理します。コンテナが検出された場合、アセンブリはキー コンテナの情報で署名されます。キー コンテナを検出できなかった場合、コンパイラは **/keyfile** で指定されたファイルを検索します。キー ファイルが見つかった場合は、そのファイル内の情報を使用してアセンブリに署名され、次のコンパイル時にキー コンテナが有効になるように、キー情報がキー コンテナに組み込まれます (**sn -i** と同様)。

キー ファイルには、公開キーだけが格納される場合もあります。

アセンブリに対する署名の詳細については、「厳密な名前付きアセンブリの作成と使用」を参照してください。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクトの [プロパティ] ページを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [署名] プロパティ ページをクリックします。
3. [厳密な名前のキー ファイルを選択してください] プロパティを変更します。

このコンパイラ オプションには、AssemblyOriginatorKeyFile を使用してプログラムでアクセスできます。

参照
その他の技術情報
Visual J# コンパイラ オプション一覧 (カテゴリ別)

/libpath (アセンブリ参照場所の指定)

/reference (メタデータのインポート) (Visual J#) オプションを使って参照されるアセンブリの場所を指定します。

```
/libpath:dir1[; dir2]
```

引数
dir1

参照先アセンブリが現在の作業ディレクトリ (コンパイラを起動したディレクトリ) または共通言語ランタイムのシステム ディレクトリに存在しない場合に、コンパイラが探すディレクトリ。

dir2

アセンブリ参照を検索するための 1 つ以上の別のディレクトリ。追加のディレクトリ名はセミコロンで区切ります。

解説

/libpath は、LIB 環境変数よりも優先されます。

コンパイラは、完全には修飾されていないアセンブリ参照を次の順序で検索します。

1. 現在の作業ディレクトリ。これは、コンパイラが起動されるディレクトリです。
2. 共通言語ランタイムのシステム ディレクトリ。
3. Visual J# のシステム ディレクトリ。
4. **/libpath** で指定したディレクトリ。
5. LIB 環境変数で指定したディレクトリ。

アセンブリ参照を指定するには、/reference (メタデータのインポート) (Visual J#) を使用します。

/libpath は追加して指定できます。繰り返して指定すると前の値に追加されます。

/libpath を使う代わりに、必要なアセンブリを作業ディレクトリにコピーする方法もあります。この場合は、単にアセンブリ名を **/reference** に渡します。その後、作業ディレクトリからアセンブリを削除できます。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [参照パス] タブをクリックします。
3. [参照パス] ボックスの内容を変更します。

使用例

`t2.jsl` をコンパイルして .exe ファイルを作成するには、次のコマンド ラインを使います。コンパイラは、作業ディレクトリと C ドライブの assemblies ディレクトリでアセンブリ参照を探します。

```
vjc /libpath:c:\assemblies /reference:t2.dll t2.jsl
```

参照
関連項目
[参照パス] ページ (プロジェクト デザイナ) (C#、J#)
その他の技術情報
Visual J# コンパイラ オプション一覧 (カテゴリ別)

/linkresource (.NET Framework リソースへのリンク) (Visual J#)

出力ファイル内に .NET Framework リソースへのリンクを作成します。リソース ファイル自体は、出力ファイルに含まれません。

```
/linkresource:filename [,identifier]
```

引数
filename

アセンブリからのリンク先である .NET Framework リソース ファイル。

identifier (省略可能)

リソースの論理名。リソースを読み込むときに使用します。既定値はファイルの名前です。リソースに対して指定できる論理名は 1 つだけです。

解説

リソース ファイルを出力ファイルに埋め込むには、/resource (出力へのリソース ファイルの埋め込み) (Visual J#) を使用します。

Visual J# コンパイラを使って作成したリンク リソースは、アセンブリ内でパブリックになります。

/linkresource では、/target:module (アセンブリに追加するモジュールの作成) (Visual J#) 以外のいずれかの /target (出力ファイル形式の指定) (Visual J#) オプションが必要です。

たとえば、*filename* が リソース ファイル ジェネレータ (Resgen.exe) によって作成された .NET Framework リソース ファイルである場合、または開発環境で作成された .NET Framework リソース ファイルである場合は、**System.Resources** 名前空間のメンバを使用してアクセスできます。詳細については、「System.Resources.ResourceManager」を参照してください。それ以外のすべてのリソースに対しては、**System.Reflection.Assembly** クラスの **GetManifestResource*** メソッドを使用して、実行時にリソースにアクセスします。

filename には任意のファイル形式を指定できます。たとえば、ネイティブ DLL をアセンブリの一部として含め、そのネイティブ DLL をグローバル アセンブリ キャッシュにインストールして、アセンブリ内のマネージ コードからアクセスできるようにすることもできます。

/linkres は **/linkresource** の省略形です。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

- このコンパイラ オプションは、Visual Studio では利用できません。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

使用例

`in.jsl` をコンパイルし、リソース ファイル `rf.resource` にリンクさせる例を次に示します。

```
vjc /linkresource:rf.resource in.jsl
```

参照
その他の技術情報
Visual J# コンパイラ

/main (main メソッドの場所の指定) (Visual J#)

コンパイルの対象に、**main** メソッドを持つ複数のクラスが存在する場合に、プログラムのエントリ ポイントとして使用する **main** メソッドが、どのクラスに存在するかを指定します。

```
/main:class
```

引数
Class

main メソッドを含むクラス。

解説

引数に指定するクラス名は、簡易名にすることも、パッケージ名を含む完全限定名にすることもできます。

このオプションは、.exe ファイルをコンパイルする場合に使用します。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [アプリケーション] タブをクリックします。
3. [スタートアップの設定] プロパティを変更します。

使用例

`ClassA.jsl` および `ClassB.jsl` をコンパイルし、**main** メソッドが `ClassA` にあることを指定するには、次のコマンド ラインを使います。

```
vjc ClassA.jsl ClassB.jsl /main:ClassA
```

参照
関連項目
[アプリケーション] ページ (プロジェクト デザイナ) (Visual Basic)
その他の技術情報
Visual J# コンパイラ オプション一覧 (カテゴリ別)

/nologo (著作権情報の非表示) (Visual J#)

コンパイラ起動時の著作権情報とコンパイル中の情報メッセージが表示されないようにします。

```
/nologo[+| -]
```

引数
±|-

/nologo+ または **/nologo** だけを指定すると、コンパイラで著作権情報が表示されなくなります。**/nologo-** を指定すると、著作権情報が表示されるようになります。このオプションを指定しない場合も、著作権情報が表示されます。

解説

このオプションは開発環境内では利用できません。このオプションを利用できるのは、コマンド ラインからコンパイルするときだけです。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

- このコンパイラ オプションは、Visual Studio では利用できません。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

参照
その他の技術情報
Visual J# コンパイラ

/nowarn (指定した警告の非表示) (Visual J#)

1 つ以上の警告について、警告を生成するコンパイラの機能を無効にします。

```
/nowarn:number1[,number2[...]]
```

引数
number1, number2

コンパイラで表示しないようにする警告の番号。

解説

警告番号が複数ある場合は、コンマで区切ります。

必要なのは、警告 ID の数値を指定することだけです。たとえば、VJS0028 を非表示にする場合は、`/nowarn:28` と指定します。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [警告の表示なし] プロパティを変更します。

参照
関連項目
[ビルド] ページ (プロジェクト デザイナ) (J#)
その他の技術情報
Visual J# コンパイラ オプション一覧 (カテゴリ別)

/optimize (最適化の有効化/無効化) (Visual J#)

コンパイラで行われる最適化の有効と無効を切り替えて、出力ファイルのサイズを小さくし、速度と効率を高めます。また、**/optimize** は、実行時にコードを最適化するように共通言語ランタイムに指示します。

```
/optimize[+ | -]
```

引数
+ | -

既定では、**/optimize-** は有効になります。出力ファイルの最適化を行わない場合は、**/optimize-** を指定します。**/optimize** の指定は、**/optimize+** の指定と同じです。

解説

/o は **/optimize** の省略形です。

/optimize オプションと /debug オプションを組み合わせることができます。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [コードの最適化] プロパティを変更します。

使用例

`t2.jsl`をコンパイルし、コンパイラの最適化を有効にするには、次のコードを使用します。

```
vjc t2.jsl /optimize
```

参照
関連項目
[ビルド] ページ (プロジェクト デザイナ) (J#)
その他の技術情報
Visual J# コンパイラ オプション一覧 (カテゴリ別)

/out (出力ファイル名の設定) (Visual J#)

出力ファイルの名前を指定します。

```
/out:filename
```

引数
filename

コンパイラで作成される出力ファイルの名前。

解説

出力ファイルの名前を指定しない場合は、次のように処理されます。

- EXE の名前は、エントリ ポイントの main メソッドを含むソース コード ファイルの名前と同じになります。
- DLL の名前は、最初のソース コード ファイルの名前と同じになります。

作成するファイルについて、フルネームと拡張子を指定します。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [アプリケーション] タブをクリックします。
3. [アセンブリ名] プロパティを変更します。

このコンパイラ オプションをコードから設定するには

- OutputFileName を参照してください。

使用例

`t1.jsl` および `t2.jsl` をコンパイルして出力ファイル `app.exe` を作成するには、次のコマンド ラインを使います。

```
vjc t1.jsl t2.jsl /out:app.exe
```

参照
関連項目
[アプリケーション] ページ (プロジェクト デザイナ) (Visual Basic)
その他の技術情報
Visual J# コンパイラ

/recurse (サブディレクトリでのソース ファイルの検索) (Visual J#)

指定のディレクトリ (*dir*) またはプロジェクト ディレクトリのすべての子ディレクトリ内のソース コード ファイルをコンパイルします。

```
/recurse:[dir\]file
```

引数
dir (省略可能)

検索を開始するディレクトリ。指定しない場合は、プロジェクト ディレクトリから検索されます。ワイルドカード文字は使用できません。

file

検索するファイル。ワイルドカード文字を使用できます。

解説

このオプションには、*dir**file* の組み合わせを 1 つだけ指定できます。*dir**file* の組み合わせをさらに指定する場合は、**/recurse** オプションを別に指定します。

/recurse を使用しなくても、ファイル名にワイルドカードを使用すると、プロジェクト ディレクトリ内で一致するすべてのファイルをコンパイルできます。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

- このコンパイラ オプションは、Visual Studio では利用できません。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

使用例

現在のディレクトリにあるすべての .jsl ソース コード ファイルをコンパイルするには、次のコマンド ラインを使います。

```
vjc *.jsl
```

`dir1\dir2` ディレクトリと各サブディレクトリにあるすべての .jsl ソース コード ファイルをコンパイルして `dir2.dll` を生成するには、次のコマンド ラインを使います。

```
vjc /target:library /out:dir2.dll /recurse:dir1\dir2\*.jsl
```

参照
その他の技術情報
Visual J# コンパイラ

/reference (メタデータのインポート) (Visual J#)

指定ファイル内のパブリック型情報をコンパイル中のプロジェクトで使用できるようにします。

```
/reference:file[;file2]
```

引数
file, file2

アセンブリ マニフェストを含む 1 つ以上のファイル。複数のファイルをインポートする場合は、コンマまたはセミコロンでファイル名を区切ります。

解説

別のアセンブリ (アセンブリ B) を参照するアセンブリ (アセンブリ A) を参照するときに、次の状況に該当する場合は、アセンブリ B も参照する必要があります。

- アセンブリ A で使う型がアセンブリ B の型を継承しているか、またはアセンブリ B のインターフェイスを実装していて、アセンブリ B の基本型や基本インターフェイスのメンバが使用される場合。
- アセンブリ B の戻り値の型やパラメータの型を持つ、フィールド、プロパティ、イベント、またはメソッドを呼び出す場合。

/libpath (アセンブリ参照場所の指定) を使用して、1 つ以上のアセンブリ参照があるディレクトリを指定します。**/libpath** に関するトピックでも、コンパイラがアセンブリを検索するディレクトリについて説明しています。

モジュールではなくアセンブリ内の型をコンパイラで認識するには、型の解決を強制する必要があります。型の解決を実行するには、たとえば、型のインスタンスを定義します。アセンブリの型名を解決する方法は他にもあります。たとえば、アセンブリの型を継承すると、コンパイラで型名が認識されます。

/r は **/reference** の省略形です。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

- [参照の追加] ダイアログ ボックス を参照してください。

使用例

ソース ファイル `input.jsl` をコンパイルし、`metad1.dll` と `metad2.dll` のメタデータをインポートして `out.exe` を生成する例を次に示します。

```
vjc /reference:metad1.dll;metad2.dll /out:out.exe input.jsl
```

参照
その他の技術情報
Visual J# コンパイラ オプション一覧 (カテゴリ別)

/resource (出力へのリソース ファイルの埋め込み) (Visual J#)

リソース ファイルを出力ファイルに埋め込みます。

```
/resource:filename[,identifier]
```

引数
filename

出力ファイルに埋め込む .NET Framework リソース ファイル。

identifier (省略可能)

リソースの論理名。リソースを読み込むときに使用します。既定値はファイルの名前です。リソースに対して指定できる論理名は 1 つだけです。

解説

/linkresource (.NET Framework リソースへのリンク) (Visual J#) を使用すると、リソースがアセンブリにリンクされ、リソース ファイルは出力ファイルに組み込まれません。

Visual J# コンパイラを使って作成したリソースは、アセンブリ内でパブリックになります。

たとえば、*filename* が リソース ファイル ジェネレータ (Resgen.exe) によって作成された .NET Framework リソース ファイルである場合、または開発環境で作成された .NET Framework リソース ファイルである場合は、System.Resources 名前空間のメンバを使用してアクセスできます。詳細については、「System.Resources.ResourceManager」を参照してください。それ以外のすべてのリソースに対しては、Assembly クラスの GetManifestResource* メソッドを使用して、実行時にリソースにアクセスします。

/res は **/resource** の省略形です。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクトにリソース ファイルを追加します。
2. ソリューション エクスプローラで、埋め込むファイルを選択します。
3. [プロパティ] ウィンドウで、リソース ファイルの [ビルド アクション] をクリックします。
4. [ビルド アクション] を [埋め込まれたリソース] に設定します。

使用例

`in.jsl` をコンパイルし、リソース ファイル `rf.resource` をアタッチする例を次に示します。

```
vjc /resource:rf.resource in.jsl
```

参照
その他の技術情報
Visual J# コンパイラ

/securescoping (パッケージ スコープを持つメンバへのアセンブリ外からのアクセスの禁止)

パッケージ スコープを持つクラスのスコープをパッケージ内に制限します。

```
/securescoping[+ | -]
```

引数
+ | -

既定では、**/securescoping-** は有効になります。パッケージ スコープを持つ項目のメタデータをパブリックとしてマークするには、**/securescoping-** を指定します。**/securescoping+** の指定は、**/securescoping** の指定と同じです。

解説

クラスはパッケージに関連付けられます。パッケージが格納されているアセンブリを別のコンパイルで参照するときにも、同じ名前のパッケージに含まれるクラスからは、パッケージ スコープのメンバにアクセスできます。

/securescoping オプションを指定すると、メンバのメタデータの出力時に、パッケージ スコープがアセンブリ スコープとして扱われます。パッケージ スコープを持つメンバには、プロテクト修飾子でマークされたメンバも含まれます。

/ss は **/securescoping** の省略形です。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [詳細] をクリックします。
4. [保護されたスコープを有効にする] をクリックします。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

使用例

ソース ファイル `input.jsl` をコンパイルし、このアセンブリが参照されたときにパッケージ型へのアクセスを禁止するには、次のコマンド ラインを使います。

```
vjc /securescoping /target:library input.jsl
```

参照
関連項目
[ビルドの詳細設定] ダイアログ ボックス (J#)
その他の技術情報
Visual J# コンパイラ

/target (出力ファイル形式の指定) (Visual J#)

/target コンパイラ オプションを指定するには、次の 4 つの形式のいずれかを使用します。

/target:exe

コンソール アプリケーション (.exe ファイル) を作成します。

/target:library

コード ライブラリを作成します。

/target:module

モジュールを作成します。

/target:winexe

Windows アプリケーション (.exe ファイル) を作成します。

/target を使った場合、.NET Framework のアセンブリ マニフェストが出力ファイルに組み込まれます。各 Visual J# プロジェクトには、アセンブリの設定がそのプロジェクトのプロパティとして格納されています。アセンブリの設定を表示または変更するには、「[署名] ページ (プロジェクト デザイナ)」を参照してください。

1 回の vjc.exe の実行で生成される出力ファイルは 1 つだけです。**/out** や **/target** のようなコンパイラ オプションを 2 回以上指定すると、コンパイラが最後に認識したオプションだけが有効になります。コンパイルされたすべてのファイルに関する情報は、マニフェストに収められます。マニフェストには、すべての出力ファイルのアセンブリ メタデータが含まれます。出力ファイルのメタデータを表示するには、MSIL 逆アセンブラ (Ildasm.exe) を使用します。

参照

その他の技術情報

Visual J# コンパイラ オプション一覧 (カテゴリ別)

/target:exe (コンソール アプリケーションの作成) (Visual J#)

実行可能なコンソール アプリケーション (EXE) をコンパイラで作成します。

```
/target:exe
```

解説

既定では、**/target:exe** オプションが有効になっています。実行可能ファイルは、.exe という拡張子で作成されます。

Windows ベースの実行可能プログラムを作成するには、/target:winexe (Windows プログラムの作成) (Visual J#) を使用します。

/out (出力ファイル名の設定) (Visual J#) オプションで特に指定しない限り、出力

ファイル名は main メソッドを含む入力ファイルの名前と同じになります。

.exe ファイルとしてコンパイルされるソース コード ファイルには、main メソッドが 1 つだけ必要です。コード内の複数のクラスに main メソッドが含まれている場合は、/main (main メソッドの場所の指定) (Visual J#) コンパイラ オプションによって、どのクラスの main メソッドを使うかを指定できます。

/t:e は **/target:exe** の省略形です。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [アプリケーション] タブをクリックします。
3. [出力の種類] ボックスの一覧の [コンソール アプリケーション] を選択します。

使用例

`in.jsl` をコンパイルし、`in.exe` を作成するには、次のいずれかのコマンド ラインを使用します。

```
vjc /target:exe in.jsl
vjc in.jsl
```

参照

関連項目

/target (出力ファイル形式の指定) (Visual J#)
[アプリケーション] ページ (プロジェクト デザイナ) (Visual Basic)

その他の技術情報

Visual J# コンパイラ オプション一覧 (カテゴリ別)

/target:library (コード ライブラリの作成) (Visual J#)

コンパイラで、実行可能ファイル (EXE) ではなくダイナミック リンク ライブラリ (DLL) を作成します。作成される DLL の拡張子は .dll です。

```
/target:library
```

解説

/out (出力ファイル名の設定) (Visual J#) オプションで特に指定しない限り、出力ファイル名は最初の入力ファイルと同じになります。

.dll ファイルをビルドする場合は、main メソッドは必要ありません。

/t:l は **/target:library** の省略形です。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [アプリケーション] タブをクリックします。
3. [出力の種類] ボックスの一覧の [クラス ライブラリ] を選択します。

使用例

`in.jsl` をコンパイルして `in.dll` を作成するには、次のコマンド ラインを使います。

```
vjc /target:library in.jsl
```

または

```
vjc /t:l in.jsl
```

参照

関連項目

/target (出力ファイル形式の指定) (Visual J#)

[アプリケーション] ページ (プロジェクト デザイナ) (Visual Basic)

その他の技術情報

Visual J# コンパイラ

/target:module (アセンブリに追加するモジュールの作成) (Visual J#)

Microsoft Intermediate Language (MSIL) 出力ファイルに、アセンブリ マニフェストが生成されないようにします。

```
/target:module
```

解説

既定では、出力ファイルの拡張子は .netmodule です。

アセンブリを持たないファイルは、.NET Framework 共通言語ランタイムで読み込むことができません。

使用例

`in.jsl` をコンパイルして `in.netmodule` を作成するには、次のコマンド ラインを使います。

```
vjc /target:module in.jsl
```

参照

関連項目

/target (出力ファイル形式の指定) (Visual J#)

[アプリケーション] ページ (プロジェクト デザイナ) (Visual Basic)

その他の技術情報

Visual J# コンパイラ オプション一覧 (カテゴリ別)

/target:winexe (Windows プログラムの作成) (Visual J#)

実行可能な Windows ベースのプログラム (EXE) をコンパイラで作成します。実行可能ファイルは、.exe という拡張子で作成されます。

```
/target:winexe
```

解説

Windows ベースのプログラムは、.NET Framework クラス ライブラリまたは Win32 API のユーザー インターフェイスを提供するプログラムです。

コンソール アプリケーションを作成するには、/target:exe (コンソール アプリケーションの作成) (Visual J#) を使用します。

/out (出力ファイル名の設定) (Visual J#) オプションで特に指定しない限り、出力ファイル名は main メソッドを含む入力ファイルの名前と同じになります。

.exe ファイルとしてコンパイルされるソース コード ファイルには、main メソッドが 1 つだけ必要です。コード内の複数のクラスに main メソッドが含まれている場合は、/main (main メソッドの場所の指定) (Visual J#) オプションによって、どのクラスの main メソッドを使うかを指定できます。

/t:w は **/target:winexe** の省略形です。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [アプリケーション] タブをクリックします。
3. [出力の種類] ボックスの一覧の [Windows アプリケーション] を選択します。

使用例

`in.jsl` をコンパイルして Windows ベースのプログラムを生成するには、次のコマンド ラインを使います。

```
vjc /target:winexe in.jsl
```

参照

関連項目

/target (出力ファイル形式の指定) (Visual J#)
[アプリケーション] ページ (プロジェクト デザイナ) (Visual Basic)

その他の技術情報

Visual J# コンパイラ オプション一覧 (カテゴリ別)

/utf8output (UTF-8 を使用したコンパイラ メッセージの表示) (Visual J#)

UTF-8 エンコーディングを使用してコンパイラ出力を表示します。

```
/utf8output[+ | -]
```

引数

+ | -

既定では、**/utf8output-** は有効になります。コンパイラ出力の表示で UTF-8 エンコーディングが使用されないようにするには、**/utf8output-** を指定します。**/utf8output** の指定は、**/utf8output+** の指定と同じです。

解説

国際対応の構成によっては、コンパイル出力がコンソールに正しく表示されないことがあります。このような構成では、**/utf8output** を使用してコンパイラ出力をファイルにリダイレクトします。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

- このコンパイラ オプションは、Visual Studio では利用できません。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

参照

その他の技術情報

Visual J# コンパイラ

/warn (警告レベルの指定) (Visual J#)

コンパイラが表示する警告レベルを指定します。

```
/warn:option
```

引数
option

ビルド中に表示させる警告レベルの最小値。次の 0 ～ 4 の値を指定できます。

警告レベル	説明
0	すべての警告メッセージの出力をオフにします。
1	重大な警告メッセージを表示します。
2	レベル 1 の警告に加えて、より重大度が低いいくつかの警告を表示します。表示される警告には、クラス メンバが非表示になっていることについての警告などがあります。 これは、コマンド ラインにおける既定の警告レベルです。
3	レベル 2 の警告に加えて、それより重大度が低いいくつかの警告を表示します。これには、常に true または false になる式に関する警告などが含まれます。
4	レベル 3 のすべての警告と、情報を提供するだけの警告を表示します。

解説

すべての警告をエラーとして扱うには、/warnaserror (警告のエラーとしての取り扱い) (Visual J#) を使います。

/w は **/warn** の省略形です。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [警告レベル] プロパティを変更します。

使用例

`in.jsl` をコンパイルし、レベル 1 の警告だけを表示するようにコンパイラに指示するには、次のコマンド ラインを使います。

```
vjc /warn:1 in.jsl
```

参照
関連項目
[ビルドの詳細設定] ダイアログ ボックス (J#)
その他の技術情報
Visual J# コンパイラ

/warnaserror (警告のエラーとしての取り扱い) (Visual J#)

すべての警告をエラーとして扱います。通常は警告としてレポートされるメッセージがエラーとしてレポートされ、ビルド処理が中断します。出力ファイルはビルドされません。

```
/warnaserror[+ | -]
```

引数
± | -

既定では、警告が出力ファイルの生成を妨げない、**/warnaserror-** が有効です。**/warnaserror** と同じ機能を持つ **/warnaserror+** は、警告をエラーとして扱います。

解説

コンパイラで表示する警告レベルを指定するには、/warn (警告レベルの指定) (Visual J#) を使います。

警告をエラーとして扱うと、コンパイラは警告レベルを区別しなくなります。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [警告をエラーとして扱う] グループで、[なし] または [すべて] をクリックします。

使用例

`in.jsl` をコンパイルし、警告を表示しないようにコンパイラに指示するには、次のコマンド ラインを使います。

```
vjc /warnaserror in.jsl
```

参照
関連項目
[ビルドの詳細設定] ダイアログ ボックス (J#)
その他の技術情報
Visual J# コンパイラ

/win32icon (Visual J#)

アプリケーションでビジュアルな効果を出したいときなど、アイコンのグラフィック リソースを挿入する場合に使用します。

```
/win32icon:filename
```

解説

指定項目 :

filename

出力ファイルに加える .ico ファイル。

解説

/win32icon オプションは、.ico ファイルを出力ファイルに挿入し、Windows エクスプローラでの出力ファイルの表示形式を指定します。.ico ファイルは、リソース コンパイラを使用して作成できます。リソース コンパイラは Visual C++ プログラムをコンパイルするときに呼び出され、.ico ファイルは .rc ファイルから作成されます。

.NET Framework のリソース ファイルの参照については、「/linkresource」、アタッチについては「/resource」を参照してください。.res ファイルのインポートについては、「/win32res (Win32 リソース ファイルのインポート)」を参照してください。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [アプリケーション] タブをクリックします。
3. [Win32 リソース ファイル] ボックスに、リソース ファイルのパスを指定します。

使用例

`in.jsl` をコンパイルし、.ico ファイル `rf.ico` をアタッチして `in.exe` を生成するには、次のコードを使用します。

```
vjc /win32icon:rf.ico in.cs
```

参照

その他の技術情報

C# コンパイラ オプション

/win32res (Win32 リソース ファイルのインポート) (Visual J#)

Win32 リソースを出力ファイルに挿入します。Win32 リソース ファイルは、Resource Compiler で作成できます。リソース コンパイラは、Visual C++ プログラムをコンパイルするときに呼び出されます。.res ファイルは .rc ファイルから作成されます。

```
/win32res:filename
```

引数
filename

出力ファイルに加えるリソース ファイル。

解説

Win32 リソースには、Windows エクスプローラでアプリケーションを識別するときに役立つ、バージョンやビットマップ (アイコン) の情報を含めることができます。**/win32res** を指定しない場合は、アセンブリのバージョンに基づくバージョン情報がコンパイラによって生成されます。

.NET Framework のリソース ファイルの参照については、「/linkresource (.NET Framework リソースへのリンク) (Visual J#)」、アタッチについては「/resource (出力へのリソース ファイルの埋め込み) (Visual J#)」を参照してください。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [アプリケーション] タブをクリックします。
3. [Win32 リソース ファイル] ボックスに、リソース ファイルのパスを指定します。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

使用例

`in.jsl` をコンパイルし、Win32 リソース ファイル `rf.res` をアタッチして `in.exe` を生成する例を次に示します。

```
vjc /win32res:rf.res in.jsl
```

参照
関連項目
[アプリケーション] ページ (プロジェクト デザイナ) (Visual Basic)
その他の技術情報
Visual J# コンパイラ

/x (言語拡張機能の無効化)

言語拡張機能を無効にします。

```
/x:[all | net]
```

引数
all

すべての言語拡張機能を無効にします。Visual J++ 6.0 の Microsoft 拡張機能だけでなく、.NET Framework に固有の新しい言語拡張機能もすべて無効になります。

net

.NET Framework に固有の言語拡張機能だけを無効にします。Visual J++ 6.0 の Microsoft 拡張機能は無効になりません。

解説

Visual J++ 6.0 における Microsoft 拡張機能の一覧については、「Visual J++ 6.0 のサポート」を参照してください。

.NET Framework 固有の言語拡張機能の一覧については、「.NET Framework をサポートする言語拡張機能」を参照してください。

Visual Studio 開発環境でこのコンパイラ オプションを設定するには

1. プロジェクト デザイナを開きます。詳細については、「方法 : プロジェクトのプロパティを設定する (C#、J#)」を参照してください。
2. [ビルド] タブをクリックします。
3. [詳細] をクリックします。
4. [すべての拡張子を無効にする] ボックスの一覧で、[すべての言語拡張を有効にする]、[.NET 拡張機能を無効にする]、または [すべての拡張子を無効にする] を選択します。

このコンパイラ オプションをコードから設定するには

- このコンパイラ オプションは、コードからは変更できません。

使用例

すべての言語拡張機能を無効にして `in.jsl` をコンパイルするには、次のコマンド ラインを使います。

```
vjc /x:all in.jsl
```

参照
関連項目
[ビルドの詳細設定] ダイアログ ボックス (J#)
その他の技術情報
Visual J# コンパイラ

Visual J# コンパイラ オプションに対応する Visual J++ コンパイラ オプション

Visual J# のコンパイラ オプションは、Visual J++® 6.0 のコンパイラ オプションとは異なります。次の表は、各コンパイラ オプションの対応を示しています。Visual J++ コンパイラ (jvc.exe) オプションの中には、Visual J# コンパイラ (vjc.exe) で使用できないオプションもあります。Visual J# コンパイラを使うときは、次の表に従ってメイクファイルを更新します。

Visual J# コンパイラ オプションでは、大文字と小文字が区別されません。省略形で使用できるオプションもあります。Visual J# コンパイラ オプションの一覧については、「Visual J# コンパイラ オプション一覧 (アルファベット順)」を参照してください。

Visual J++ コンパイラ オプション	**Visual J# コンパイラ オプション**	**コメント**
/?	**/?** または **/help**	使用方法を表示します。
/cp < *classpath* >	**/libpath**	コンパイル用のクラス パスを指定します。 クラス パスを指定するには、**/libpath** または LIB 環境変数を使います。 コンパイル時には、**/reference** オプションを使って、クラスが含まれているアセンブリも参照する必要があります。 **/libpath** は、LIB 環境変数よりも優先されます。
/cp:p[-]	**/libpath**	クラス パスにパスを付加します。 LIB 環境変数で指定されたパスよりも優先させるには、**/libpath** を使います。
/cp:o[-]	適用できません。	クラス パスを表示します。
/d < *directory* >	サポートされていません。	クラス ファイルを出力するルート ディレクトリを指定します。 出力ディレクトリと出力ファイル名を指定するには、**/out** を使います。
/D < *symbol* >	**/define:<symbol>**	プリプロセッサ シンボルを定義します。 **/define:** または省略形の **/d:** を使用できます。Visual J# のオプションでは大文字と小文字が区別されないため、**/D:** も使用できます。
/g	**/debug**	完全なデバッグ情報を生成するように、コンパイラに指示します。 代わりに、**/debug** オプションを使います。
/g:l または **/g:t**	**/debug:{full|pdbonly}**	生成するデバッグ情報の種類を指定します。 生成するデバッグ情報を指定します。**/debug:full** を指定すると、実行中のプログラムにデバッガをアタッチできます。
/nologo	**/nologo**	変更はありません。
/nowarn	**/warn:0**	警告を無効にします。 警告レベル 0 を指定すると、警告はすべて無効になります (省略形は **/w:0**)。
/nowrite	サポートされていません。	何も出力せず、コンパイルだけを行います。
/O	**/optimize**	最適化を有効にします。 代わりに、**/optimize** オプションを使います。

/O:J および **/O:I**	サポートされていません。	最適化の種類を指定します。
/ref	サポートされていません。	参照先クラスを再コンパイルして、最新版にします。
/verbose	サポートされていません。	コンパイルの進行状況に関する情報を出力します。
/w{0-4}	**/warn:{0-4}**	警告レベルを設定します。 **/warn: option** (省略形は **/w:{0-4}**) を使用できるようになりました。
/wx	**/warnaserror**	警告をエラーとして扱います。 代わりに、**/warnaserror** オプションを使います。
/x	**/x:all** **/x:net**	拡張機能を無効にします。 **/x:all** を使用して、すべての言語拡張機能を無効にできるようになりました。**/x:net** では、.NET Framework 固有の言語拡張機能だけが無効になります。

参照

その他の技術情報

Visual J# コンパイラ

コンパイラ エラー VJS0001 から VJS9999

ここでは、vjprvj コンパイラによって生成されるエラーについて説明します。特定のエラー メッセージに関するヘルプを表示するには、[出力] ウィンドウでエラー番号をクリックして **F1** キーを押すか、または [インデックス] の [検索する文字列] ボックスにエラー番号を入力します。

参照

その他の技術情報

Visual J# コンパイラ

コンパイラ エラー VJS1001

エラー メッセージ
ソース ファイルが指定されていません。

コマンド ラインで Visual J# コンパイラが起動され、1 つ以上のコンパイラ オプションが指定されていましたが、ソース コード ファイルが指定されていませんでした。1 つ以上のソース コード ファイルを指定してください。

コンパイラ エラー VJS1002

エラー メッセージ
'<オプション名>' は不明なオプションです。

スラッシュ (/) で始まる値が Visual J# コンパイラに渡されましたが、有効なコンパイラ オプションではありません。コンパイラ オプションについては、「Visual J# コンパイラ オプション」を参照してください。

次の例では、VJS1002 エラーが生成されます。

```
// VJS1002.jsl
// compile with: /nolog
// VJS1002 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1003

エラー メッセージ
オプション '<オプション名>' は、'+'、'-'、または nothing に先行しなければなりません。

ブール値 (または値なし) を受け入れるコンパイラ オプションに無効な値が渡されました。

次の例では、VJS1003 エラーが生成されます。

```
// VJS1003.jsl
// compile with: /optimize:0
// VJS1003 expected
// to resolve, use /optimize (not /optimize:0)
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1004

エラー メッセージ
オプション '<オプション名>' は、':' と値に先行しなければなりません。

指定されたコンパイラ オプションの形式が不正です。コンパイラ オプションの構文については、「Visual J# コンパイラ オプション」を参照してください。

次の例では、VJS1004 エラーが生成されます。

```
// VJS1004.jsl
// compile with: /define+
// VJS1004 expected
public class MyClass
{
   public static void main()
   {
    }
}
```

コンパイラ エラー VJS1005

エラー メッセージ
ファイル 'ファイル' が見つかりません。

ファイル名がコンパイラに渡されましたが、コンパイラはそのファイルを見つけることができませんでした。コンパイラは、現在のディレクトリのファイルを検索します。サブディレクトリのファイルを検索するには、/recurse コンパイラ オプションを使います。

コンパイラ エラー VJS1006

エラー メッセージ
ソース ファイル '<ファイル名>' を読み取れません。

Visual J# コンパイラに渡されたソース コード ファイルには、読み取りアクセスが指定されていません。

コンパイラ エラー VJS1009

エラー メッセージ
/reference ファイル '<ファイル名>' が見つかりません。

ファイルが /reference コンパイラ オプションに渡されましたが、コンパイラはそのファイルを見つけることができませんでした。

次の例では、VJS1009 エラーが生成されます。

```
// VJS1009.jsl
// compile with: /reference:a_file_you_do_not_have.dll
// VJS1009 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1010

エラー メッセージ
/reference ファイル '<ファイル名>' はアセンブリではありません。

ファイルが /reference コンパイラ オプションに渡されましたが、そのファイルにはアセンブリ マニフェストが含まれていませんでした。

使用例

```
// VJS1010a.cpp
// compile with: /LD
// C++ source code
class SomeClass {};
```

```
// VJS1010b.jsl
// compile with: /reference:VJS1010a.dll /target:library
// VJS1010 expected
public class MyClass {}
```

コンパイラ エラー VJS1011

エラー メッセージ
'@' はレスポンス ファイル名に先行しなければなりません。

応答ファイルを指定するコマンド ライン オプションに、ファイル名が含まれていません。詳細については、「@ (応答ファイルの指定)」を参照してください。

次の例では、VJS1011 エラーが生成されます。

```
// VJS1011.jsl
// compile with: @
// VJS1011 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1012

エラー メッセージ
応答ファイル '<ファイル名>' を読み取れません。

Visual J# コンパイラに渡された応答ファイルには、読み取りアクセスが指定されていません。

コンパイラ エラー VJS1013

エラー メッセージ
レスポンス ファイル '<ファイル名>' が 2 回以上見つかりました。

応答ファイルが、同じコンパイルで 2 回指定されました。コンパイルで指定できる応答ファイルは 1 つだけです。詳細については、「@ (応答ファイルの指定)」を参照してください。

次の例では、VJS1013 エラーが生成されます。

```
// VJS1013.jsl
// compile with: @x.rsp @x.rsp
// VJS1013 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1014

エラー メッセージ
レスポンス ファイル '<ファイル名>' の引用符に、閉じかっこがありません。

レスポンス ファイルで始まりの引用符が見つかりましたが、終わりの引用符がありません。以下に示すサンプルでは、次の内容のレスポンス ファイルがディスクに保存されている場合を想定しています。

```
/target:library "
```

次の例では、VJS1014 エラーが生成されます。

```
// VJS1014.jsl
// compile with: @VJS1014.rsp
// VJS1014 expected
public class MyClass
{
}
```

コンパイラ エラー VJS1016

エラー メッセージ
'<文字列>' は /target の有効な値ではありません。

無効なパラメータが /target コンパイラ オプションに渡されました。

次の例では、VJS1016 エラーが生成されます。

```
// VJS1016.jsl
// compile with: /target:exec
// VJS1016 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1017

エラー メッセージ
'<文字列>' は /x の有効な値ではありません。

無効なパラメータが /x コンパイラ オプションに渡されました。

次の例では、VJS1017 エラーが生成されます。

```
// VJS1017.jsl
// compile with: /x:module
// VJS1017 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1019

エラー メッセージ
/baseaddress の値 '<値>' が不適切です。

無効な値が /baseaddress コンパイラ オプションに渡されました。

次の例では、VJS1019 エラーが生成されます。

```
// VJS1019.jsl
// compile with: /target:library /baseaddress:r
// VJS1019 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1020

エラー メッセージ
/warn の値は 0-4 の範囲になければなりません。

無効な値が /warn コンパイラ オプションに渡されました。

次の例では、VJS1020 エラーが生成されます。

```
// VJS1020.jsl
// compile with: /warn:5
// VJS1020 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1022

エラー メッセージ
ファイル '<ファイル名>' に書き込めません。

同じ名前のファイルがコンピュータで使用されているため、コンパイルで指定された出力ファイルを作成できませんでした。出力ファイルの名前を変更するか、同名のファイルを閉じてください。

コンパイラの警告 (レベル 1) VJS1023

エラー メッセージ
'<クラス名>' の重複する main を無視します。/main:<classname> を使用してエントリ ポイントを指定してください。

複数の **main** メソッドが見つかりました。最初の **main** メソッドはプログラムのエントリ ポイントとして使用され、それ以降の **main** メソッドは無視されます。

次の例では、VJS1023 エラーが生成されます。

```
// VJS1023.jsl
// compile with: /W:1
public class MyClass
{
   public static void main()
   {
   }
}

public class MyClass2   // VJS1023
{
   // this main method is ignored
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1024

エラー メッセージ
/main オプションが不適切です : クラス '<クラス名>' は適切な main メソッドを持っていません。

/main コンパイラ オプションで指定されたクラスには、プログラムのエントリ ポイントとして使用できる **main** メソッドがありません。**main** メソッドを持つクラスを指定してください。

次の例では、VJS1024 エラーが生成されます。

```
// VJS1024.jsl
// compile with: /main:MyClass
public class MyClass  // VJS1024, specify /main:MyClass2 instead
{
}

public class MyClass2
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1025

エラー メッセージ
/main オプションで名づけられたクラス '<クラス名>' がコンパイルで見つかりません。

/main コンパイラ オプションで指定されたクラスがソース コードで見つかりません。コンパイルするソース コード ファイルのいずれかに存在するクラスを指定してください。

次の例では、VJS1025 エラーが生成されます。

```
// VJS1025.jsl
// compile with: /main:MyClass2
// VJS1025 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1026

エラー メッセージ
/target は 'exe' ですが、適切な main メソッドがコンパイルで見つかりません。

プログラムの実行時にエントリ ポイントとして機能する **main** メソッドが見つかりませんでした。

次の例では、VJS1026 エラーが生成されます。

```
// VJS1026.jsl
// compile with: /target:exe
// VJS1026 expected
public class MyClass
{
   // uncomment the following lines to resolve
   // public static void main()
   // {
   // }
}
```

コンパイラ エラー VJS1027

エラー メッセージ
/target は 'winexe' ですが、適切な main メソッドがコンパイルで見つかりません。

プログラムの実行時にエントリ ポイントとして機能する **main** メソッドが見つかりませんでした。

次の例では、VJS1027 エラーが生成されます。

```
// VJS1027.jsl
// compile with: /target:winexe
// VJS1027 expected
public class MyClass
{
   // uncomment the following lines to resolve
   // public static void main()
   // {
   // }
}
```

コンパイラ エラー VJS1028

エラー メッセージ
main が指定されている場合は、/target は 'exe' または 'winexe' でなければなりません。

/target:library コンパイラ オプションが、/main コンパイラ オプションと矛盾しています。

実行可能ファイルを生成する場合は **/target:exe** または **/target:winexe** を指定します。ライブラリを生成する場合は **/main** オプションを削除します。

次の例では、VJS1028 エラーが生成されます。

```
// VJS1028.jsl
// compile with: /target:library /main:MyClass
// VJS1028 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1029

/resource または /linkresource コマンド ライン オプションでは、1 つのリソース識別子のみ指定できます。

/linkresource コンパイラ オプションと /resource コンパイラ オプションを使用すると、アセンブリごとに複数のリソース識別子を指定できます。ただし、**/resource** フラグは、識別子ごとに別々に指定する必要があります。

次の例では、VJS1029 エラーが生成されます。

```
// VJS1029.jsl
// compile with: /res:VJS1029.resource,name1,name2
// VJS1029 expected
// try /res:VJS1029.resource /res:name1.resource /res:Name2.resource
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1031

エラー メッセージ
エラーとして処理された警告

コンパイル時に /warnaserror オプションを使う場合は、すべての警告がエラーとして扱われます。

次の例では、VJS1031 エラーが生成されます。

```
// VJS1031.jsl
// compile with: /warnaserror /W:3
// VJS1031 expected
class MyClass
{
   public static void main()
   {
      int i;   // warning
   }
}
```

コンパイラ エラー VJS1032

エラー メッセージ
'<値>' は /debug に対して有効な値ではありません : 有効な値は full または pdbonly のみです。

/debug コンパイラ オプションの指定が無効です。

次の例では、VJS1032 エラーが生成されます。

```
// VJS1032.jsl
// compile with: /debug:0
// VJS1032 expected
// to resolve, use /debug (not /debug:0)
public class MyClass
{
}
```

コンパイラ エラー VJS1033

エラー メッセージ
/resource または /linkresource コマンド ライン オプションを指定して、リソース ファイル名を指定してください。

/resource コンパイラ オプションまたは /linkresource コンパイラ オプションの指定が無効です。

次の例では、VJS1033 エラーが生成されます。

```
// VJS1033.jsl
// compile with: /resource:
// VJS1033 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1034

エラー メッセージ
'<番号>' は、/nowarn コマンド ライン オプションに対して有効な警告番号ではありません。

/nowarn コンパイラ オプションに渡された値は、有効な警告番号ではありませんでした。

次の例では、VJS1034 エラーが生成されます。

```
// VJS1034.jsl
// compile with: /nowarn:0000 /target:library
// VJS1034 expected
public class MyClass
{
}
```

コンパイラ エラー VJS1035

エラー メッセージ
'<オプション名>' コマンド ライン オプションに対するファイル指定がありません。

コンパイラ オプションにファイルを指定してください。

次の例では、VJS1035 エラーが生成されます。

```
// VJS1035.jsl
// compile with: /recurse: /target:library
// VJS1035 expected
// try the following compiler options instead
// /recurse:*.jsl /target:library
public class MyClass
{
}
```

コンパイラ エラー VJS1036

エラー メッセージ
リソース識別子 '<名前>' は、このアセンブリで既に使用されています。

ユーザー定義のリソース ファイル名は、アセンブリで既に使用されています。

次の例では、VJS1036 エラーが生成されます。

```
// VJS1036.jsl
// compile with: /res:VJS1036A.resources,a /res:VJS1036B.resources,a
// VJS1036 expected
public class MyClass
   {
   }
```

コンパイラ エラー VJS1037

エラー メッセージ
競合するオプションが指定されました : 警告をエラーとして扱います。警告レベルは 0 です。

同じコンパイルで /warnaserror と /warn:0 の両方を指定できません。

次の例では、VJS1037 エラーが生成されます。

```
// VJS1037.jsl
// compile with: /W:0 /warnaserror
// VJS1037 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1038

エラー メッセージ
コード ページ '<コード ページ名>' は無効か、またはコンピュータにインストールされていません。

Visual J# コンパイラを呼び出すときに、/codepage オプションに不正な値が指定されました。

コンパイラ エラー VJS1039

エラー メッセージ
モジュールをビルド中にリソース ファイルをリンクすることはできません。

同じコンパイルで /target:module (アセンブリに追加するモジュールの作成) と /linkresource (.NET Framework リソースへのリンク) の両方は指定できません。

コンパイラ エラー VJS1040

エラー メッセージ
ファイル 'filename' に書き込めません -- 'reason'

表示されている原因によって、コンパイラがファイルに書き込むことができません。たとえば、MSIL 逆アセンブラ (Ildasm.exe) でファイルを開き、vjc.exe を使ってファイルを再度ビルドしようとすると、このエラーが生成されます。

コンパイラ エラー VJS1041

エラー メッセージ
'<オプション名>' コマンド ライン オプションの値を指定してください。

指定されたコンパイラ オプションの形式が不正です。コンパイラ オプションの構文については、「Visual J# コンパイラ オプション」を参照してください。

次の例では、VJS1041 エラーが生成されます。

```
// VJS1041.jsl
// compile with: /define:
// VJS1041 expected
public class MyClass
{
   public static void main()
   {
    }
}
```

コンパイラ エラー VJS1042

エラー メッセージ
インクリメンタル ビルドは /out で指定されたターゲット名が必要です。

インクリメンタル ビルドを実行する場合は、/out オプションで出力ファイルを指定する必要があります。

このエラーを解決するには

- 出力ファイルの名前を指定します。

使用例
次の例では、VJS1042 エラーが生成されます。

```
public class MyClass
{
   public static void main()
   {
    }
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1044

エラー メッセージ
ファイル 'ファイル' から読み取れません

このファイル名は、ハード ディスク上の有効なファイルではない可能性があります。デバイスまたはコンソールに対応付けられている可能性があります。

サンプル :

vjc /res:con.res MyClass.jsl

con.res はディスクに存在しないファイル、MyClass.jsl は有効な J# ファイルです。

コンパイラ エラー VJS1076

エラー メッセージ
Unexpected end of file

コードの構造を解析し終える前に、ファイルの最後に到達しました。ファイルの終端付近をチェックし、すべてのコードが完結していることを確認してください。たとえば、右中かっこ (}) が欠落していることなどが考えられます。

このエラーは、コンパイラ エラー VJS1118 として報告される場合があります。

コンパイラ エラー VJS1077

エラー メッセージ
未終了のコメントです。

複数行のコメントが終了していません。

次の例では、VJS1077 エラーが生成されます。

```
// VJS1077.jsl
class MyClass
{
/*   // VJS1077, no end to comment
   public static void main()
   {
      System.out.println("test");
   }
}
```

コンパイラ エラー VJS1078

エラー メッセージ
予期しない文字 '<文字>' です。

予期しない文字が原因で構文エラーが発生しました。

次の例では、VJS1078 エラーが生成されます。

```
// VJS1078.jsl
class MyClass
{
   @   // VJS1078
}
```

コンパイラ エラー VJS1079

エラー メッセージ
'<トークン>' は有効なキーワードまたは識別子ではありません。

Visual J# では、**const** と **goto** は有効な識別子ではありません。

次の例では、VJS1079 エラーが生成されます。

```
// VJS1079.jsl
class MyClass
{
   int goto;   // VJS1079
}
```

コンパイラ エラー VJS1080

エラー メッセージ
#error '<ユーザー定義エラー>'

このエラーは、**#error** ディレクティブが実行されたときに発生します。

次の例では、VJS1080 エラーが生成されます。

```
// VJS1080.jsl
public class MyClass
{
   public static void main()
   {
      #error A user-defined error  // VJS1080
   }
}
```

コンパイラの警告 (レベル 1) VJS1081

エラー メッセージ
#warning '<ユーザー定義の警告>'

この警告は、**#warning** ディレクティブが実行されたときに発生します。

次の例では、VJS1081 エラーが生成されます。

```
// VJS1081.jsl
// compile with: /W:1
public class MyClass
{
   public static void main()
   {
      #warning A user-defined warning  // VJS1081
   }
}
```

コンパイラ エラー VJS1082

エラー メッセージ
条件付きディレクティブは行でスペース以外の最初の文字でなければなりません。

条件付きディレクティブが、行の他のテキストの後に指定されました。条件付きディレクティブは、必ず行の最初のトークンとして指定してください。

次の例では、VJS1082 エラーが生成されます。

```
// VJS1082.jsl
#define Xc
public class MyClass
{
   public static void main()
   {
      int i; #if X  // VJS1082
         int i;
      #endif
   }
}
```

コンパイラ エラー VJS1083

エラー メッセージ
条件付き識別子が必要ですが、見つかったのは '<文字列>' です。

不正な形式のシンボルが条件付きディレクティブで検出されました。

次の例では、VJS1083 エラーが生成されます。

```
// VJS1083.jsl
#define -X   // VJS1083
#define VALID_ID   // OK
class MyClass
{
}
```

コンパイラ エラー VJS1084

エラー メッセージ
条件付きディレクティブが必要ですが、見つかったのは '<文字列>' です。

不正な形式の条件付きディレクティブが検出されました。

次の例では、VJS1084 エラーが生成されます。

```
// VJS1084.jsl
#-define X   // VJS1084
class MyClass
{
}
```

コンパイラ エラー VJS1085

エラー メッセージ
条件付き識別子が必要です。

不正な形式の条件付きディレクティブが検出されました。

次の例では、VJS1085 エラーが生成されます。

```
// VJS1085.jsl
#define   // VJS1085
class MyClass
{
}
```

コンパイラ エラー VJS1086

エラー メッセージ
条件付きディレクティブが必要ですが、行の最後に達しました。

行末に到達しましたが、識別子は見つかりませんでした。

次の例では、VJS1086 エラーが生成されます。

```
// VJS1086.jsl
class MyClass
{
   #   // VJS1086
   #if DIRECTIVE   // OK
   #endif
}
```

コンパイラ エラー VJS1087

エラー メッセージ
#line ディレクティブの後には、行番号または 'default' が必要です。

不完全な #line ディレクティブが検出されました。

次の例では、VJS1087 エラーが生成されます。

```
// VJS1087.jsl
public class MyClass
{
   public static void main()
   {
      #line testing   // VJS1087
      // try the following line instead
      // #line 21
    }
}
```

コンパイラ エラー VJS1088

エラー メッセージ
'#<テキスト>' は条件付きディレクティブではありません。

シャープ記号 (#) で始まるトークンが検出されましたが、有効な条件付きディレクティブではありません。

次の例では、VJS1088 エラーが生成されます。

```
// VJS1088.jsl
public class MyClass
{
   public static void main()
   {
      #linee 22  // VJS1088
      // try the following line instead
      // #line 22
   }
}
```

コンパイラ エラー VJS1089

エラー メッセージ
条件付きディレクティブの後に余分なテキストがあります。

整形式の条件付きディレクティブの後に別のテキストが追加されていました。

次の例では、VJS1089 エラーが生成されます。

```
// VJS1089.jsl
public class MyClass
{
   public static void main()
   {
      #line 22 77   // VJS1089
   }
}
```

コンパイラ エラー VJS1090

エラー メッセージ
ソースの後に #define または #undef を持てません。

ファイルのソース コードを開始した後に、**#define** ディレクティブまたは **#undef** ディレクティブが見つかりました。**#define** ディレクティブと **#undef** ディレクティブは、ソースの最初のトークンの前に指定する必要があります。

次の例では、VJS1090 エラーが生成されます。

```
// VJS1090.jsl
import System.*;
#define DEBUG  // VJS1090, try moving this line before import statement
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1091

エラー メッセージ
#if が '#<ディレクティブ>' と一致しません。

条件付きディレクティブ **#endif** または **#else** が見つかりましたが、対応する **#if** ディレクティブがありません。

次の例では、VJS1091 エラーが生成されます。

```
// VJS1091.jsl
public class MyClass
{
   public static void main()
   {
      // #if DEBUG
      #endif   // VJS1091, uncomment the #if statement to resolve
   }
}
```

コンパイラ エラー VJS1092

エラー メッセージ
#if に対応する #endif がありません。

#if ディレクティブが見つかりましたが、対応する **#endif** ディレクティブがありません。

次の例では、VJS1092 エラーが生成されます。

```
// VJS1092.jsl
public class MyClass
{
   public static void main()
   {
      #if DEBUG   // VJS1092
      // uncomment the following line to resolve
      // #endif
   }
}
```

コンパイラ エラー VJS1093

エラー メッセージ
'#if' は複数の '#else' を持っています。

#if ステートメントに指定できる **#else** ディレクティブは、多くても 1 つです。

次の例では、VJS1093 エラーが生成されます。

```
// VJS1093.jsl
public class MyClass
{
   public static void main()
   {
      #if DEBUG
         // DEBUG defined
      #else
         // DEBUG not defined
      #else   // VJS1093, second #else not valid
         // something wrong
      #endif
   }
}
```

コンパイラ エラー VJS1094

エラー メッセージ
条件式の行末が不適切です。

条件付きディレクティブの形式が不正です。

次の例では、VJS1094 エラーが生成されます。

```
// VJS1094.jsl
public class MyClass
{
   public static void main()
   {
      #if   // VJS1094, add an expression to evaluate
      // try the following line instead
      // #if DEBUG
      #endif
   }
}
```

コンパイラ エラー VJS1095

エラー メッセージ
条件式に閉じかっこのない '(' があります。

左かっこに対応する右かっこがないため、条件式の形式が不正です。

次の例では、VJS1095 エラーが生成されます。

```
// VJS1095.jsl
public class MyClass
{
   public static void main()
   {
      #if (DEBUG   // VJS1095
      // try the following line instead
      // #if (DEBUG)
      #endif
   }
}
```

コンパイラ エラー VJS1096

エラー メッセージ
定義済みの 'true' または 'false' は変更できません。

ブール定数の true または false を使ってシンボルを定義しようとしました。true または false は変更できません。

次の例では、VJS1096 エラーが生成されます。

```
// VJS1096.jsl
#define true   // VJS1096, don't use true as a user-defined symbol
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1097

エラー メッセージ
.NET 拡張機能が無効のときは、ディレクティブ '#<ディレクティブ名>' はサポートされません。

.NET Framework 拡張機能を無効にした場合は、一部の処理ディレクティブが無効になります。詳細については、/x コンパイラ オプションに関するトピックを参照してください。

次の例では、VJS1097 エラーが生成されます。

```
// VJS1097.jsl
// compile with: /x:net
public class MyClass
{
   public static void main()
   {
      #line 32   // VJS1097
   }
}
```

コンパイラ エラー VJS1118

エラー メッセージ
予期しない '*トークン*' です。

予期しないトークンが見つかりました。一部の特殊なトークンについては、より具体的なエラーが生成され、VJS1118 として報告される場合があります。たとえば、右中かっこ (}) が省略されていると、コンパイラから予期しない EOF トークンとして報告されます。この場合、VJS1118 として報告されたエラーは、実際にはコンパイラ エラー VJS1076 になります。

次の例では、VJS1118 エラーが生成されます。

```
// VJS1118.jsl
public class MyClass
{
   public static void main(String[] args)
   {
      )   // VJS1118
   }
}
```

コンパイラ エラー VJS1119

エラー メッセージ
'<トークン>' が必要です。

ソース コードを解析できませんでした。

次の例では、VJS1119 エラーが生成されます。

```
// VJS1119.jsl
package test
public class MyClass // VJS1119, end previous line with semicolon
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1120

エラー メッセージ
'class'、'interface'、または 'delegate' が必要ですが、'<トークン>' が見つかりました。

無効な宣言が見つかりました。

このエラーを解決するには

1. 無効な宣言を探します。
2. 宣言を修正します。

使用例
次の例では、VJS1120 エラーが生成されます。

```
public class Program
{
   public static void main(String[] args)
   {
   }
   class MyClass
   {
      public void Test()
         {
            abstract int t;   // VJS1120
            int x = 10;   // OK
         }
   }
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1122

エラー メッセージ
型が必要ですが、'void' が見つかりました。

void は、メソッドの定義または宣言のパラメータとして有効な型ではありません。

次の例では、VJS1122 エラーが生成されます。

```
// VJS1122.jsl
class MyClass
{
    public void Test(void t)   // VJS1122
   // try the following line instead
   // public void Test(int t)
   {
   }
}
```

コンパイラ エラー VJS1123

エラー メッセージ
不適切なメンバ宣言です。

クラス メンバが正しく宣言されていませんでした。

次の例では、VJS1123 エラーが生成されます。

```
// VJS1123.jsl
public class MyClass
{
   public static void main()
   {
   }
   xxx class MyClass2   // VJS1123, xxx is not a valid access modifer
   // try the following line instead
   // public class MyClass2
   {
   }
}
```

コンパイラ エラー VJS1124

エラー メッセージ
@attribute.return は、メソッド宣言だけで有効です。

@attribute.return 属性が、対象外の構成要素に適用されました。この属性を適用できるのはメソッドだけです。

次の例では、VJS1124 エラーが生成されます。

```
// VJS1124.jsl
import System.Runtime.InteropServices.*;
/** @attribute.return MarshalAs(UnmanagedType.Error) */   // VJS1124
public class MyClass
{
   // OK
   /** @attribute.return MarshalAs(UnmanagedType.Error) */
   public int Test()
   {
      return -1;
   }
}
```

コンパイラ エラー VJS1125

エラー メッセージ
属性宣言の終わりが不適切です。

属性の宣言が不完全です。

次の例では、VJS1125 エラーが生成されます。

```
// VJS1125.jsl
/** @attribute System.CLSCompliant */   // VJS1125
// try the following line instead
// /** @attribute System.CLSCompliant(true) */
class MyClass
{
}
```

コンパイラ エラー VJS1126

エラー メッセージ
@attribute.method は、メソッド宣言だけで有効です。

@attribute.method 属性が、対象外の構成要素に適用されました。この属性を適用できるのはメソッドだけです。

次の例では、VJS1126 エラーが生成されます。

```
// VJS1126.jsl
/** @attribute.method System.ObsoleteAttribute("Hello") */   // VJS1126
public class MyClass
{
   // following line is OK
   /** @attribute.method System.ObsoleteAttribute("Hello") */
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1127

エラー メッセージ
ユーザー定義のコンストラクタは、ComImportAttribute に設定されているクラスでは使用できません。

ユーザー定義のコンストラクタは、**ComImportAttribute** でマークされているクラスでは指定できません。**ComImport** クラスのコンストラクタは、共通言語ランタイムの COM 相互運用層に用意されています。COM オブジェクトを作成するのに必要なこのコンストラクタは、ランタイムのマネージ オブジェクトのようになります。

次の例では、VJS1127 エラーが生成されます。

```
// VJS1127.jsl
import System.Runtime.InteropServices.*;
/** @attribute ComImport() */
/** @attribute Guid("fb6363f0-721b-478e-bfc6-bc30f7b06be5") */
public class MyClass   // VJS1127
{
   // To solve the problem, remove the constructor below.
   public MyClass(){}
}
```

コンパイラ エラー VJS1128

エラー メッセージ
ComImportAttribute に設定されているクラスには、GuidAttribute を指定しなければなりません。

GuidAttribute は **ComImportAttribute** でマークされているクラスでプリセットされる必要があります。そうでない場合、.NET Framework の .NET COM 相互運用層は、COM クラスを作成できません。

次の例では、VJS1128 エラーが生成されます。

```
// VJS1128.jsl
// compile with: /target:library
import System.Runtime.InteropServices.*;
/** @attribute ComImport() */
// add the following line to resolve
// /** @attribute Guid("00000000-0000-0000-0000-000000000001") */
public class MyClass   // VJS1128
{
}
```

コンパイラ エラー VJS1129

エラー メッセージ
StructLayoutAttribute(LayoutKind.Explicit) に設定されているクラスのインスタンス フィールド '<フィールド名>' には、FieldOffsetAttribute を指定しなければなりません。

StructLayout **Explicit** が有効である場合、フィールドは FieldOffset でマークされる必要があります。

次の例では、VJS1129 エラーが生成されます。

```
// VJS1129.jsl
// compile with: /target:library
import System.Runtime.InteropServices.*;

/** @attribute StructLayout(LayoutKind.Explicit, Size = 4)*/
class Myclass
{
   int a;   // VJS1129
}
```

コンパイラ エラー VJS1153

エラー メッセージ
クラス '<クラス名>' は自らを継承しています。

クラスが自身を基本クラスとして使うことは無効です。

次の例では、VJS1153 エラーが生成されます。

```
// VJS1153.jsl
public class MyClass extends MyClass   // VJS1153
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1156

エラー メッセージ
スーパークラス '<クラス名>' は final であるため、拡張できません。

final とマークされているクラスは、基本クラスとして使用できません。

次の例では、VJS1156 エラーが生成されます。

```
// VJS1156.jsl
final class MyClass2
{
   public static void main()
   {
   }
}

public class MyClass extends MyClass2   // VJS1156
{
}
```

コンパイラ エラー VJS1157

エラー メッセージ
インターフェイス '<インターフェイス名>' は拡張できません。'implements' を使用してください。

インターフェイスをクラスによって拡張することはできませんが、実装することはできます。

次の例では、VJS1157 エラーが生成されます。

```
// VJS1157.jsl
interface Point
{
}

public class MyClass extends Point   // VJS1157
// try the following line instead
// public class MyClass implements Point
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1158

エラー メッセージ
インターフェイスでインターフェイスを実装することはできません。'extends' を使用してください。

インターフェイスを別のインターフェイスによって実装することはできませんが、拡張することはできます。

次の例では、VJS1158 エラーが生成されます。

```
// VJS1158.jsl
interface Point
{
}

interface Point2 implements Point   // VJS1158
// try the following line instead
// interface Point2 extends Point
{
}

public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1159

エラー メッセージ
インターフェイス '<インターフェイス名>' は 2 回実装されています。

インターフェイスは、1 つの型につき 1 回だけ実装できます。

次の例では、VJS1159 エラーが生成されます。

```
// VJS1159.jsl
interface Point
{
}

interface Point2 extends Point, Point   // VJS1159, Point listed twice
// try the following line instead
// interface Point2 extends Point
{
}

public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1161

エラー メッセージ
クラス '<クラス名>' が見つかりません。

型のインポートが指定されましたが、コンパイルに型が指定されていません。このエラーを修正するには、クラスのコードが含まれたソース ファイルをコマンド ラインまたはプロジェクトに追加するか、適切な参照を追加します。

使用例

```
// VJS1161a.jsl
// compile with: /target:library
public class Class1 {}
```

```
// VJS1161b.jsl
// compile with: /target:library
import Class1;   // VJS1161 add VJS1161a.jsl to compilation
class MyClass {}
```

コンパイラ エラー VJS1163

エラー メッセージ
'<クラス名>' はインターフェイスではありません。

クラスは実装できませんが、拡張できます。

次の例では、VJS1163 エラーが生成されます。

```
// VJS1163.jsl
interface Point
{
}

public class MyClass
{
}

public class MyClass2 implements MyClass   // VJS1163
// try the following line instead
// public class MyClass2 extends MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1164

エラー メッセージ
インターフェイス宣言はブロックの中では使用できません。

インターフェイスが、有効でない場所で宣言されました。

次の例では、VJS1164 エラーが生成されます。

```
// VJS1164.jsl
// compile with: /target:library
class MyClass
{
   {
      interface I   // VJS1164, interface declaration not allowed here
      {
      }

      class C   // type declarations allowed
      {
      }
   }
}
```

コンパイラ エラー VJS1166

エラー メッセージ
static および instance 初期化子はインターフェイスで使用できません。

インターフェイスに static 初期化子ブロックや instance 初期化子ブロックを含めることはできませんが、メンバの初期化を含めることはできます。

次の例では、VJS1166 エラーが生成されます。

```
// VJS1166.jsl
// compile with: /target:library
interface I1
{
   static   // VJS1166 static initializer block in interface
   {
      int i = 10;
   }

   static int x = 0;   // OK: static member initializer
}

interface I2
{
   {   // VJS1166 instance initalizer block in interface
      int i = 10;
   }

   int x = 20;   // OK: instance member initializer
}
```

コンパイラ エラー VJS1167

エラー メッセージ
コンストラクタはインターフェイスで使用できません。

インターフェイスにコンストラクタの宣言または定義を含めることはできません。

次の例では、VJS1167 エラーが生成されます。

```
// VJS1167
public interface Point
{
   Point()   // VJS1167, constructor not allowed
   {
   }
}

public class MyClass
{
   public MyClass ()   // OK, Ctors allowed in class declarations
   {
   }

   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1169

エラー メッセージ
'<クラス名>' を抽象として宣言するか、または '<継承クラス名>.<メンバ名>' を実装しなければなりません。

クラス (<クラス名>) は、abstract として宣言する場合以外は、継承クラス (基本クラス) の抽象メンバを実装する必要があります。

次の例では、VJS1169 エラーが生成されます。

```
// VJS1169.jsl
public interface Point
{
   void SetPoint();
}

public class MyClass implements Point   // VJS1169
{
   // uncomment member definition to resolve, or declare MyClass abstract
   // public void SetPoint()
   // {
   // }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1170

エラー メッセージ
'<クラス名>' を抽象として宣言するか、または次のメソッドを実装しなければなりません : <メソッド名>

クラス (<クラス名>) は、abstract として宣言する場合以外は、継承クラスのすべての抽象メンバを実装する必要があります。

次の例では、VJS1170 エラーが生成されます。

```
// VJS1170.jsl
public interface Point
{
   void SetPoint();
   int GetPoint();
}

public class MyClass implements Point   // VJS1170
// to resolve, either implement methods in Point or
// declare MyClass as abstract
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1172

エラー メッセージ
内部クラス '<内部クラス名>' をインスタンス化するには、外部クラス '<クラス名>' の明示的なインスタンスが必要です。

非静的な内部クラス メンバのインスタンスを作成するには、外側のクラスのオブジェクト インスタンスが必要です。

次の例では、VJS1172 エラーが生成されます。

```
// VJS1172.jsl
class MyClass
{
   public static void main (String [] args)
   {
      new Class1.InnerClass();   // VJS1172
      // to resolve, define InnerClass as static
      // or try the following line instead
      // new Class1 (). new InnerClass ();
   }
}

class Class1
{
   public class InnerClass
   {
   }
}
```

コンパイラ エラー VJS1173

エラー メッセージ
内部クラス '<内部クラス名>' のコンストラクタを呼び出すには、外部クラス '<クラス名>' の明示的なインスタンスが必要です。

静的コンテキストで内部クラスの作成を試みました。内部クラスを作成するには、インスタンスが必要です。

次の例では、VJS1173 エラーが生成されます。

```
// VJS1173a.jsl
// compile with: /target:library
class MyClass
{
   public class InnerClass
   {
   }
   public static class Inner2 extends InnerClass
   {
      Inner2 ()
      {
         super();   // VJS1173, make Inner2 an instance class
      }
   }
}
```

すべてのクラス コンストラクタにはスーパー クラスの呼び出しが含まれているため、インスタンスが必要です。

次の例では、VJS1173 エラーが生成されます。

```
// VJS1173b.jsl
// compile with: /target:library
public class MyClass extends Class1.Inner
{
   public MyClass()   // VJS1173, Class1.Inner needs to be static
   {
   }
}

public class Class1
{
   public class Inner
   {
   }
}
```

コンパイラ エラー VJS1174

エラー メッセージ
型はそれを囲むクラス 'クラス' と同じ簡易名を持っています。

内部の型と外部クラスの名前が同じです。このエラーを解決するには、いずれかの型の名前を変更します。

次の例では、VJS1174 エラーが生成されます。

```
// VJS1174.jsl
public class MyClass
{
   public static void main()
   {
   }
   public class MyClass   // VJS1174
   // try the following line instead
   // public class InnerClass
   {
   }
}
```

コンパイラ エラー VJS1175

エラー メッセージ
内部クラス '<クラス名>' は静的な初期化子を持っています。

インスタンスの内部クラスに静的初期化子を含めることはできません。ただし、静的内部クラスには、静的初期化子を含めることができます。

次の例では、VJS1175 エラーが生成されます。

```
// VJS1175.jsl
public class MyClass
{
   public static void main()
   {
   }

   public class InnerClass   // VJS1175
   {
      static int i = 0;
      // try the following line instead
      // int i = 0;
   }
}
```

コンパイラ エラー VJS1176

エラー メッセージ
フィールド '<フィールド名>' を内部クラスで static として宣言できません。

インスタンスの内部クラスに static フィールドを含めることはできません。ただし、static 内部クラスには、static フィールド宣言を含めることができます。

次の例では、VJS1176 エラーが生成されます。

```
// VJS1176.jsl
public class MyClass
{
   public static void main()
   {
   }

   public class InnerClass
   {
      static int i = 0;   // VJS1176
      // try the following line instead
      // int i = 0;
   }
}
```

コンパイラ エラー VJS1177

エラー メッセージ
メソッド '<メソッド名>' を内部クラスで static に宣言できません。

インスタンスの内部クラスに static メソッドを含めることはできません。ただし、static 内部クラスには、static メソッド宣言を含めることができます。

次の例では、VJS1177 エラーが生成されます。

```
// VJS1177.jsl
public class MyClass
{
   public static void main()
   {
   }

   public class InnerClass
   {
      public static int Test()   // VJS1177, delete static modifier
      {
      }
   }
}
```

コンパイラ エラー VJS1179

エラー メッセージ
内部クラスでインターフェイス '<インターフェイス名>' を宣言できません。

非静的内部クラスにインターフェイスを含めることはできません。

次の例では、VJS1179 エラーが生成されます。

```
// VJS1179.jsl
class MyClass
{
   public class Inner
   {
      public interface InnerInterface   // VJS1179, to resolve, declare Inner as static
      {
      }
   }
   public static void main (String [] args)
   {
   }
}
```

コンパイラ エラー VJS1180

エラー メッセージ
修飾された 'new' は、'<名前>' ではなく、単純型名を使用する必要があります。

内部クラスでオブジェクトを作成する場合は、内部型の名前を単純型名にする必要があります。

次の例では、VJS1180 エラーが生成されます。

```
// VJS1180.jsl
class MyClass
{
   public static void main(String [] args)
   {
      new Class1 ().new Class1.Inner();   // VJS1180
      // try the following line instead
      // new Class1 (). new Inner();
   }
}

class Class1
{
   public class Inner
   {
   }
}
```

コンパイラ エラー VJS1182

エラー メッセージ
'内部クラス' を囲むインスタンスとして this を 'クラス' に割り当てることはできません。

内部クラスのインスタンスを作成するには、内部クラスの外側のクラスのインスタンスが必要です。

次の例では、VJS1182 エラーが生成されます。

```
// VJS1182.jsl
class MyClass
{
   public void Test()
   {
      new Class1.Inner();    // VJS1182
      // try the following line instead
      // new Class1 (). new Inner ();
   }
   public static void main (String [] args)
   {
   }
}

class Class1
{
   public class Inner
   {
   }
}
```

コンパイラ エラー VJS1183

エラー メッセージ
別のソース ファイルで宣言された型 '<型>' の再宣言です。

同名の型が、同じコンパイルの 2 つのソース コード ファイルで宣言されています。

次の例では、VJS1183 エラーが生成されます。

```
// VJS1183a.jsl
// compile with: /target:library
public class MyClass
{
}
```

```
// VJS1183b.jsl
// compile with: VJS1183a.jsl
public class MyClass   // VJS1183, MyClass also declared in VJS1183a.jsl
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1184

エラー メッセージ
同じソース ファイル内で、型 '<型>' が複数回宣言されました。

同名の型が、ソース コード ファイルで 2 回宣言されています。

次の例では、VJS1184 エラーが生成されます。

```
// VJS1184.jsl
public class MyClass
{
   public static void main()
   {
   }
}

class MyClass   // VJS1184
{
}
```

コンパイラ エラー VJS1185

エラー メッセージ
型 '<型>' は、少なくとも 2 つの 'import on demand' ステートメントによってインポートされました。

1 つの型名が 2 つの異なるパッケージに含まれており、2 つの **import** ステートメントによってインポートされています。

使用例

```
// VJS1185a.jsl
// compile with: /target:library
package pkg1;
public class MyType {}
```

```
// VJS1185b.jsl
// compile with: /target:library
package pkg2;
public class MyType {}
```

次の例では、VJS1185 エラーが生成されます。

```
// VJS1185c.jsl
// compile with: VJS1185a.jsl VJS1185b.jsl
// VJS1185 expected
import pkg1.*;
import pkg2.*;

class MyClass
{
   MyType t;
}
```

コンパイラの警告 (レベル 1) VJS1186

エラー メッセージ
型 '<型>' が、<インポート> と java.lang の両方によってインポートされました。java.lang 型を使用します。

特に指定しない限り、java.lang パッケージはインポートされます。この警告は、java.lang パッケージのクラスが別のパッケージで定義されており、**import** ステートメントを使ってインポートされることを示します。

次の例では、VJS1186 エラーが生成されます。

```
// VJS1186.jsl
// compile with: /W:1
// VJS1186 expected
import System.*;
class MyClass
{
   public static void main (String [] args)
   {
      Exception e;  // VJS1186: Exception is in java.lang package and .NET framework System
package
   }
}
```

コンパイラ エラー VJS1190

エラー メッセージ
フィールド '<フィールド名>' があいまいです。このフィールドは、複数のスーパークラスまたはスーパーインターフェイスから継承されています。

派生クラスのフィールドの使用を限定する必要があります。

次の例では、VJS1190 エラーが生成されます。

```
// VJS1190.jsl
interface I1
{
   int i = 0;
}

interface I2
{
   int i = 0;
}

public class MyClass implements I1, I2
{
   public static void main()
   {
      int j = i;   // VJS1190
      // try the following line instead
      // int j = I1.i;
   }
}
```

コンパイラ エラー VJS1191

エラー メッセージ
フィールド '<フィールド名>' は、現在のクラスで既に定義されています。

フィールドがクラスで 2 回以上定義されています。

次の例では、VJS1191 エラーが生成されます。

```
// VJS1191.jsl
public class MyClass
{
   int i;
   int i;   // VJS1191, delete line or rename identifier
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1192

エラー メッセージ
空の最終フィールド '<フィールド名>' は、宣言またはコンストラクタで初期化されていません。

final とマークされているフィールドは、宣言時またはコンストラクタ内で初期化する必要があります。

次の例では、VJS1192 エラーが生成されます。

```
// VJS1192.jsl
public class MyClass   // VJS1192
{
   final int i;
   // try the following line instead
   // final int i = 0;
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1193

エラー メッセージ
最終フィールド '<フィールド名>' は 2 回以上初期化されています。

final フィールドは、2 回以上初期化できません。**final** フィールドが、インスタンス初期化ブロックとコンストラクタで初期化されました。

次の例では、VJS1193 エラーが生成されます。

```
// VJS1193.jsl
class MyClass   // VJS1193, delete one of the initializations
{
   final int i;
   {
      i =  10;
   }
   MyClass ()
   {
      i = 20;
   }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1194

エラー メッセージ
最終変数またはフィールド '<フィールド名>' を割り当てられません。

final フィールドが宣言で初期化された後、コンストラクタで初期化されました。

次の例では、VJS1194 エラーが生成されます。

```
// VJS1194.jsl
class MyClass
{
   final int i = 20;
   MyClass ()
   {
      i = 10;   // VJS1194
   }
   public static void main (String [] args)
   {
   }
}
```

コンパイラ エラー VJS1195

エラー メッセージ
空の最終フィールド '<フィールド名>' は既に初期化されています。

final フィールドがコンストラクタで 2 回初期化されました。

次の例では、VJS1195 エラーが生成されます。

```
// VJS1195.jsl
class MyClass
{
   final int i;
   MyClass ()
   {
      i = 10;
      i = 20;   // VJS1195, delete one of the initializations
   }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1196

エラー メッセージ
空の最終フィールド '<フィールド名>' は、Loop ステートメント内では初期化できません。

final フィールドの初期化がループ内に指定されています。**final** フィールドを初期化できるのは 1 度だけです。

次の例では、VJS1196 エラーが生成されます。

```
// VJS1196.jsl
class MyClass
{
   final int i;
   {
      int k = 1;
      while (k == 1)
      {
         i = 10;   // VJS1196, in a loop
      }
      i = 10;
   }

   public static void main (String [] args)
   {
   }
}
```

コンパイラ エラー VJS1197

エラー メッセージ
代入式の左側には左辺値を指定しなければなりません。

代入の形式が不正です。

次の例では、VJS1197 エラーが生成されます。

```
// VJS1197.jsl
public class MyClass
{
   public static void main()
   {
      int i = 9;
      0 = i;   // VJS1197
      // try the following line instead
      // i = 0;
   }
}
```

コンパイラ エラー VJS1200

エラー メッセージ
型 '<型>' は、現在のコンテキストで既に知られている型です。

同名の 2 つの型が同じスコープで宣言されています。

次の例では、VJS1200 エラーが生成されます。

```
// VJS1200.jsl
class MyClass
{
   public class Type1
   {
   }
   public class Type1   // VJS1200, second type with same name
   {
   }
}
```

コンパイラ エラー VJS1201

エラー メッセージ
名前 '<識別子>' は、現在のコンテキストで定義されていません。

使用されている識別子が現在のスコープで定義されていません。

次の例では、VJS1201 エラーが生成されます。

```
// VJS1201.jsl
public class MyClass
{
   public static void main()
   {
      // int ii = 0;
      ii++;   // VJS1201, uncomment previous line to resolve
   }
}
```

コンパイラ エラー VJS1202

エラー メッセージ
変数 '<変数名>' は、現在のブロックで既に定義されています。

同名の変数が 2 回宣言されています。

次の例では、VJS1202 エラーが生成されます。

```
// VJS1202.jsl
public class MyClass
{
   public static void main()
   {
      int i;
      int i;   // VJS1202, delete or rename
   }
}
```

コンパイラ エラー VJS1203

エラー メッセージ
ローカル変数 '<変数名>' は、使用前に初期化されていません。

変数は、使う前に初期化する必要があります。

次の例では、VJS1204 エラーが生成されます。

```
// VJS1203.jsl
public class MyClass
{
   public static void main()
   {
      char c;
      // try the following line instead
      // char c = 'a';
      if (c == 'a')   // VJS1203
         ;
   }
}
```

コンパイラ エラー VJS1204

エラー メッセージ
if 条件は、'type' ではなく、boolean でなければなりません。

if ステートメントの条件部分がブール式ではありません。

次の例では、VJS1204 エラーが生成されます。

```
// VJS1204.jsl
public class MyClass
{
   public static void main()
   {
      char c = 'a';
      if ('a')   // VJS1204
      // try the following line instead
      // if (c == 'a')
         ;
   }
}
```

コンパイラ エラー VJS1205

エラー メッセージ
for 条件は、'<型>' ではなく、boolean でなければなりません。

for ステートメントの条件部分がブール式ではありません。

次の例では、VJS1205 エラーが生成されます。

```
// VJS1205.jsl
public class MyClass
{
   public static void main()
   {
      for (int i = 0 ; 'c' ; i++)   // VJS1205
      // try the following line instead
      // for (int i = 0 ; i < 10 ; i++)
         ;
   }
}
```

コンパイラ エラー VJS1206

エラー メッセージ
while 条件は、'<型>' ではなく、boolean でなければなりません。

while ステートメントの条件について説明しているセクションがブール式ではありません。

次の例では、VJS1206 エラーが生成されます。

```
// VJS1206.jsl
public class MyClass
{
   public static void main()
   {
      char c = 'a';
      while ('c')   // VJS1206
      // try the following line instead
      // while (c != 'c')
         ;
   }
}
```

コンパイラ エラー VJS1207

エラー メッセージ
do 条件は、'<型>' ではなく、boolean でなければなりません。

do ループの条件付きステートメントは、ブール型に評価される必要があります。

次の例では、VJS1207 エラーが生成されます。

```
// VJS1207.jsl
public class MyClass
{
   public static void main()
   {
      int i = 0;
      do
      {
         i++;
      } while ('c');   // VJS1207
      // try the following line instead
      // } while (i < 10);
   }
}
```

コンパイラ エラー VJS1208

エラー メッセージ
ラベル '<ラベル名>' は現在のステートメント階層で既に定義されています。

同名のラベルがラベル スコープで宣言されています。

次の例では、VJS1208 エラーが生成されます。

```
// VJS1208.jsl
class MyClass
{
   public static void main()
   {
      int i = 0;
      label1:
      {
         i = 1;
         label1:   // VJS1208, already in label1 scope
         {
            i = 1;
         }
      }
   }
}
```

コンパイラ エラー VJS1209

エラー メッセージ
ラベル '<ラベル名>' は現在のステートメント階層で定義されていません。

ラベルが呼び出されましたが、宣言されていませんでした。

次の例では、VJS1209 エラーが生成されます。

```
// VJS1209.jsl
class MyClass
{
   public static void main (String [] args)
   {
      for ( int i = 0 ; i < 10 ; i++ )
      {
         continue MyLabel;   // VJS1209, no such label declared
      }
   }
}
```

コンパイラ エラー VJS1210

エラー メッセージ
この既定のセレクタは、switch に 1 つしか使用できません。

switch ステートメントで既定の条件が 2 つ以上見つかりました。

次の例では、VJS1210 エラーが生成されます。

```
// VJS1210.jsl
public class MyClass
{
   public static void main()
   {
      int i = 0;
      switch (i)
      {
         case 0:
         default:
         default:   // VJS1210
      }
   }
}
```

コンパイラ エラー VJS1211

エラー メッセージ
ラベル '<ラベル名>' は現在のスイッチで既に定義されています。

重複した値を持つ 2 つの **case** ステートメントが見つかりました。

次の例では、VJS1211 エラーが生成されます。

```
// VJS1211.jsl
public class MyClass
{
   public static void main()
   {
      int i = 0;
      switch (i)
      {
         case 0:
         case 0:   // VJS1211
         // try the following line instead
         // case 1:
      }
   }
}
```

コンパイラ エラー VJS1212

エラー メッセージ
switch ラベルは '<型>' ではなく、序数でなければなりません。

case ステートメントに指定された値が序数ではありませんでした。

次の例では、VJS1212 エラーが生成されます。

```
// VJS1212.jsl
public class MyClass
{
   public static void main()
   {
      int i = 0;
      switch (i)
      {
         case "abc":   // VJS1212, string not an ordinal
         // try the following line instead
         // case 0:
         default:
      }
   }
}
```

コンパイラ エラー VJS1213

エラー メッセージ
switch ラベルは定数値でなければなりません。

case ステートメントに渡された値が定数ではありませんでした。

次の例では、VJS1213 エラーが生成されます。

```
// VJS1213.jsl
public class MyClass
{
   public static void main()
   {
      int i = 0;
      switch (i)
      {
         case i:   // VJS1213
         // try the following line instead
         // case 0:
         default:
      }
   }
}
```

コンパイラ エラー VJS1214

エラー メッセージ
switch 式は序数でなければなりません。

switch ステートメントに渡された値が序数ではありませんでした。

次の例では、VJS1214 エラーが生成されます。

```
// VJS1214.jsl
public class MyClass
{
   public static void main()
   {
      int i = 0;
      switch ("abc")   // VJS1214, string not an ordinal
      // try the following line instead
      // switch (i)
      {
         case 0:
         default:
      }
   }
}
```

コンパイラ エラー VJS1215

エラー メッセージ
synchronized の式は参照型でなければなりません。

synchronized 式に渡す値は、参照型であることが必要です。

次の例では、VJS1215 エラーが生成されます。

```
// VJS1215.jsl
class Test
{
   public static void main()
   {
      boolean tf = true;
      Test t = new Test();
      synchronized(tf)   // VJS1215, to resolve, pass t to synchronized
      {
         synchronized(tf)
         {
         }
      }
   }
}
```

コンパイラ エラー VJS1216

エラー メッセージ
ステートメントは制御範囲外にあります。

実行されないステートメントが検出されました。

次の例では、VJS1216 エラーが生成されます。

```
// VJS1216.jsl
class Test
{
   public static void main()
   {
      int i = 1;
      do
      {
         break;
         i++;   // VJS1216, break statement makes this unreachable
      } while (i != 0);
   }
}
```

コンパイラ エラー VJS1217

エラー メッセージ
このステートメントはアクションを実行しません。

無効なステートメントが検出されました。

次の例では、VJS1217 エラーが生成されます。

```
// VJS1217.jsl
class Test
{
   public static void main()
   {
      int i = 1;
      i;   // VJS1217, no effect
   }
}
```

コンパイラ エラー VJS1219

エラー メッセージ
このコンテキストには、型名 '型' ではなく、式が必要です。

式を必要とする場所で型名が見つかりました。

次の例では、VJS1219 エラーが生成されます。

```
// VJS1219.jsl
public class MyClass
{
   public static void main()
   {
      int i = 0;
      switch (MyClass)   // VJS1219
      // try the following line instead
      // switch (i)
      {
         case 0:
         default:
      }
   }
}
```

コンパイラ エラー VJS1220

エラー メッセージ
名前 '<名前>' を解決できません。

メンバへの参照が見つかりましたが、この型の参照先メンバが存在しません。一般に、このエラーは、プロパティを参照するための構文が正しくない場合に発生します。他の .NET 言語と異なり、J# では、"get_" 形式と "set_" 形式のプロパティ アクセサを使用する必要があります。

次の 2 つの例では、VJS1220 が生成されます。

```
// VJS1220.vjs
class MyClass {
   public static void main() {
      System.Console.WriteLine(System.IntPtr.Size);   // VJS1220
      System.Console.WriteLine(System.IntPtr.get_Size());   // OK
   }
}
```

```
// VJS1220.jsl
public class MyClass
{
   /*
   public static class InnerClass
   {
      public static void Test()
      {
      }
   }
   */
   public static void main (String [] args)
   {
      MyClass.InnerClass.Test();   // VJS1220, uncomment InnerClass to resolve
   }
}
```

コンパイラ エラー VJS1222

エラー メッセージ
loop コンテキスト外で continue ステートメントを実行することはできません。

ループの外側で **continue** ステートメントが検出されました。

次の例では、VJS1222 エラーが生成されます。

```
// VJS1222.jsl
public class MyClass
{
   public static void main()
   {
      continue;   // VJS1222
   }
}
```

コンパイラ エラー VJS1223

エラー メッセージ
'<型>' にメソッド '<メソッド名>' が見つかりません。

メソッドのシグネチャがメソッドの呼び出しと一致していません。

次の例では、VJS1223 エラーが生成されます。

```
// VJS1223.jsl
public class MyClass
{
   public void Test()
   {
   }

   public static void main()
   {
      MyClass x = new MyClass();
      x.Test(1);   // VJS1223
      // try the following line instead
      // x.Test();
   }
}
```

コンパイラ エラー VJS1224

エラー メッセージ
クラス '<クラス名>' で、super への呼び出しを限定できません。スーパー クラスは内部クラスではありません。

スーパー クラスは内部クラスではないため、**super** の呼び出しを解決できませんでした。

次の例では、VJS1224 エラーが生成されます。

```
// VJS1224.jsl
public class MyClass
{
   public MyClass ()
   {
      new Class1 ().super ();   // VJS1224, delete this line to resolve
   }
   public static void main (String [] args)
   {
   }
}

public class Class1
{
}
```

コンパイラ エラー VJS1225

エラー メッセージ
クラス '<クラス名>' で、super への呼び出しを限定修飾する必要があります。スーパー クラスは内部クラスです。

型によって非静的内部クラスが拡張される場合、スーパー クラスのコンストラクタを直接呼び出すことはできません。内部クラスの外側のクラスのインスタンスが必要です。

次の例では、VJS1225 エラーが生成されます。

```
// VJS1225.jsl
public class MyClass extends Class1.Inner
{
   MyClass()
   {
      super();   // VJS1225
      // try the following line to resolve
      // new Class1(). super();
   }
   public static void main (String [] args)
   {
   }
}

public class Class1
{
   public class Inner
   // or, make the inner class static
   // public static class Inner
   {
   }
}
```

コンパイラ エラー VJS1227

エラー メッセージ
コンストラクタ '<コンストラクタ名>' が見つかりません。

コンストラクタのシグネチャがコンストラクタの呼び出しと一致していません。

次の例では、VJS1227 エラーが生成されます。

```
// VJS1227.jsl
public class MyClass
{
   // Uncomment this constructor to fix the error.
   // MyClass(int i)
   // {
   // }
   public static void main()
   {
      MyClass x = new MyClass(9);   // VJS1227
      x.Test();
   }
   public void Test()
   {
   }
}
```

コンパイラ エラー VJS1228

エラー メッセージ
メソッド '<メソッド名>' は、現在のクラスで既に定義されています。

メソッドが 2 回定義されています。

次の例では、VJS1228 エラーが生成されます。

```
// VJS1228.jsl
public class MyClass
{
   public void Test()
   {
   }
   public static void main()
   {
   }
   public void Test()   // VJS1228, duplicate method definition
   {
   }
}
```

コンパイラ エラー VJS1231

エラー メッセージ
メソッド '<メソッド名>' の呼び出しがあいまいです。

どのメソッドを呼び出すかを判断できませんでした。

次の例では、VJS1231 エラーが生成されます。

```
// VJS1231.jsl

public class MyClass1
{
}

public class MyClass2 extends MyClass1
{
}

class MyClass3
{
   public void Test(MyClass2 mc2, MyClass1 mc1)
   {
   }
   public void Test(MyClass1 mc1, MyClass2 mc2)
   {
   }
   public static void main()
   {
      MyClass2 mc2 = new MyClass2();
      Test(mc2, mc2);   // VJS1231
   }
}
```

コンパイラ エラー VJS1232

エラー メッセージ
メソッド '<メソッド名>' への明示的な呼び出しを行えません。

コンパイラによって生成される内部メソッドを明示的に呼び出すことはできません。

コンパイラ エラー VJS1233

エラー メッセージ
メソッド名が不適切です。

メソッド名が無効です。識別子には次の文字を使用できます。

- 文字 A ～ Z および a ～ z
- 記号 _ および $
- 数字 0 ～ 9

次の例では、VJS1233 エラーが生成されます。

```
// VJS1233.jsl
public class MyClass
{
   public static void main (String [] args)
   {
      1234();   // VJS1233, invalid method name
   }
}
```

コンパイラ エラー VJS1234

エラー メッセージ
明示的な 'this' または 'super' の呼び出しを行うことができるのは、コンストラクタの最初のステートメントだけです。

this および **super** は、コンストラクタで使う場合、先頭行だけで使用できます。

次の例では、VJS1234 エラーが生成されます。

```
// VJS1234.jsl
public class MyClass1
{
   MyClass1()
   {
   }
}

public class MyClass2 extends MyClass1
{
   int i;
   MyClass2()
   {
      i = 0;
      super();   // VJS1234, make this the first line
   }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1237

エラー メッセージ
例外 '<例外名>' はキャッチされません。throws 句に記述されていません。

throws 句に、メソッドからスローされた例外か、**try-catch** 句でキャッチされた例外を指定する必要があります。

次の例では、VJS1237 エラーが生成されます。

```
// VJS1237.jsl
// compile with: /target:library
public class MyClass
{
   // Try the following line instead:
   // void method() throws Exception
   void method ()
   {
      throw new Exception ();   // VJS1237
      // Exception should be caught or declared using throws clause
   }
}
```

コンパイラ エラー VJS1238

エラー メッセージ
例外 '<例外名>' を、静的初期化子、または静的フィールド初期化子によってスローすることはできません。

チェックされた (非ランタイム) 例外は、静的初期化子ブロックでスローできません。

次の例では、VJS1238 エラーが生成されます。

```
// VJS1238.jsl
class MyClass
{
   static
   {
      throw new java.io.IOException();   // VJS1238
      // OK to throw run-time exceptions
      // throw new NullPointerException();
   }
}
```

コンパイラ エラー VJS1239

エラー メッセージ
フィールド初期化子またはコンストラクタによってスローされた例外 '<例外名>' はキャッチされません。コンストラクタの throws 句に記述されていません。

コンストラクタが例外をスローしましたが、例外は宣言されていませんでした。

次の例では、VJS1239 エラーが生成されます。

throw ステートメントを使ってメソッドから例外をスローする場合は、**throws** 句を使ってメソッド宣言で例外を宣言する必要があります。

```
// VJS1239a.jsl
public class MyClass
{
   MyClass() // throws Exception
   {
      throw new Exception();   // VJS1239, uncomment throws clause in declaration
   }
   public static void main (String [] args) throws Exception
   {
      new MyClass ();
   }
}
```

throws ステートメントを使ってメソッドから例外をスローする場合は、**try-catch** ブロックで例外をキャッチする必要があります。

```
// VJS1239b.jsl
public class MyClass   // VJS1239, uncomment the following lines to resolve
{
//   {
//      try
//      {
         int x = Test();
//      } catch (Exception e) {
//      }
//   }
   int Test() throws Exception
   {
      return 10;
   }
   public static void main (String [] args)
   {
      new MyClass ();
   }
}
```

コンパイラ エラー VJS1240

エラー メッセージ
インスタンス メソッドで静的メソッド '<メソッド名>' をオーバーライドすることはできません。

オーバーライドするメソッドとオーバーライドされるメソッドで使うインスタンスおよび静的なステータスは、同じである必要があります。

次の例では、VJS1240 エラーが生成されます。

```
// VJS1240.jsl
class MyClass1
{
   public static void Test()
   {
   }
}

class MyClass2 extends MyClass1
{
   public void Test()   // VJS1240, static/instance must agree with MyClass1 Test
   // try the following line instead
   // public static void Test()
   {
   }
   public static void main()
   {
      MyClass2 x = new MyClass2();
      x.Test();
   }
}
```

コンパイラ エラー VJS1241

エラー メッセージ
静的メソッドで、インスタンス メソッド '<メソッド名>' をオーバーライドすることはできません。

オーバーライドするメソッドとオーバーライドされるメソッドで使うインスタンスおよび静的なステータスは、同じである必要があります。

次の例では、VJS1241 エラーが生成されます。

```
// VJS1241.jsl
class MyClass1
{
   public void Test()
   {
   }
}

class MyClass2 extends MyClass1
{
   public static void Test()   // VJS1241, static/instance must agree with MyClass1 Test
   // try the following line instead
   // public void Test()

   {
   }
   public static void main()
   {
      MyClass2 x = new MyClass2();
      x.Test();
   }
}
```

コンパイラ エラー VJS1242

エラー メッセージ
final メソッド '<メソッド名>' をオーバーライドすることはできません。

final メソッドはオーバーライドできません。

次の例では、VJS1242 エラーが生成されます。

```
// VJS1242.jsl
// compile with: /target:library
public class Base
{
   public static final int Test()
   {
      return 0;
   }
}

public class Derived extends Base
{
   public static int Test()   // VJS1242, hides Base.Test
   {
      return 1;
   }
}
```

コンパイラ エラー VJS1243

エラー メッセージ
final メソッド '<メソッド名>' をオーバーライドすることはできません。

final メソッドはオーバーライドできません。

次の例では、VJS1243 エラーが生成されます。

```
// VJS1243.jsl
class MyClass1
{
   public final void Test()
   {
   }
}

class MyClass2 extends MyClass1
{
   public void Test()   // VJS1243, MyClass1.Test is final
   {
   }
   public static void main()
   {
      MyClass2 x = new MyClass2();
      x.Test();
   }
}
```

コンパイラ エラー VJS1245

エラー メッセージ
メソッド '<メソッド名 1>' および '<メソッド名 2>' は戻り値の型のみが異なります。

オーバーライドするメソッドとオーバーライドされるメソッドの戻り値の型は、同じである必要があります。

次の例では、VJS1245 エラーが生成されます。

```
// VJS1245.jsl
class MyClass1
{
   public void Test()
   {
   }
}

class MyClass2 extends MyClass1
{
   public int Test()   // VJS1245, MyClass1.Test returns void
   {
   }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1246

エラー メッセージ
メソッド '<メソッド名>' は、スーパー メソッドの throws 句にない '<例外名>' をスローします。

オーバーライドするメソッドとオーバーライドされるメソッドの例外の指定が、一致していません。

次の例では、VJS1246 エラーが生成されます。

```
// VJS1246.jsl
public class MyClass extends Base
{
   public void Method() throws Exception   // VJS1246
   // uncomment exception specification in base method to resolve
   {
   }
   public static void main (String [] args)
   {
   }
}

class Base
{
   public void Method() // throws Exception
   {
   }
}
```

コンパイラ エラー VJS1247

エラー メッセージ
アクセス制限がより強いメソッド '<メソッド名>' を使って、スーパー メソッドを再定義しようとしています。

オーバーライドするメソッドに、オーバーライドされるメソッドよりも低いアクセス レベルを指定できません。

次の例では、VJS1247 エラーが生成されます。

```
// VJS1247.jsl
public class MyClass1
{
   public void Text()
   {
   }
}

public class MyClass2 extends MyClass1
{
   private void Text()   // VJS1247, base class Text is public
   {
   }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1248

エラー メッセージ
メソッド 'メソッド' にアクセスできません。

メソッドにアクセスできませんでした。

次の例では、VJS1248 エラーが生成されます。

```
// VJS1248.jsl
public class MyClass1
{
   private void Test()
   {
   }
}

class MyClass2 extends MyClass1
{
   public static void main()
   {
      MyClass1 x = new MyClass1();
      x.Test();   // VJS1248, Test is private
   }
}
```

コンパイラ エラー VJS1249

エラー メッセージ
このオブジェクトを通して、継承されたメソッド '<メンバ名>' にアクセスできません。

ユーザー補助レベルのため、<メンバ名> にアクセスできません。

使用例

```
// VJS1249a.jsl
// compile with: /target:library
package p;
public class MyClass
{
   protected void m() {}
   public void n() {}
}
```

次の例では、VJS1249 エラーが生成されます。

```
// VJS1249b.jsl
// compile with: VJS1249a.jsl
public class MyClass2 extends p.MyClass
{
   public static void main(String[] args)
   {
      new p.MyClass().m();   // VJS1249
      new p.MyClass().n();   // OK
   }
}
```

コンパイラ エラー VJS1250

エラー メッセージ
メソッド '<メソッド名>' のプレフィックスは、'<型>' ではなく、参照でなければなりません。

メソッド呼び出しは、メソッドが含まれている型への参照によって修飾される必要があります。

次の例では、VJS1250 エラーが生成されます。

```
// VJS1250.jsl
public class MyClass
{
   public static void main (String [] args)
   {
      MyClass x = new MyClass();
      int i;
      i = 10;
      i.Test();   // VJS1250
      // try the following line instead
      // x.Test();
   }
   public void Test()
   {
   }
}
```

コンパイラ エラー VJS1251

エラー メッセージ
メソッド '<メソッド名>' の戻り値がありません。

メソッドの戻り値の型が指定されていません。

次の例では、VJS1251 エラーが生成されます。

```
// VJS1251.jsl
public class MyClass
{
   public static main()   // VJS1251
   // try the following line instead
   // public static void main()
   {
   }
}
```

コンパイラ エラー VJS1252

エラー メッセージ
ByRef 'パラメータ' に対応する実際のパラメータは割り当て可能でなければなりません。

参照渡しのパラメータを使うメソッドが呼び出されましたが、渡されたパラメータにはリテラルを代入できません。変数を渡してください。

使用例

```
// VJS1252a.cs
// compile with: /target:library
// a C# program
public class RefClass
{
   public void method (ref int x) {}
}
```

```
// VJS1252b.jsl
// compile with: /reference:VJS1252a.dll
public class MyClass
{
   public static void main (String [] args)
   {
      int x = 10;
      new RefClass ().method (10);   // VJS1252
      new RefClass ().method (x);   // OK
   }
}
```

コンパイラ エラー VJS1253

エラー メッセージ
型 '<型 1>' のオブジェクトを、型 '<型 2>' のパラメータとして渡せません。

参照パラメータが渡されましたが、メソッドで宣言された型と一致していません。

使用例

```
// VJS1253a.cs
// compile with: /target:library
// a C# program
public class RefClass
{
   public void Test(ref int x) {}
}
```

次の例では、VJS1253 エラーが生成されます。

```
// VJS1253b.jsl
// compile with: /reference:VJS1253a.dll
class MyClass
{
   public static void main (String [] args)
   {
      short x = 0;
      new RefClass ().Test(x);   // VJS1253

      int y = 0;
      new RefClass ().Test(y);   // OK
   }
}
```

コンパイラ エラー VJS1254

エラー メッセージ
インターフェイスで定義されたメソッド '<メソッド名>' に本体を指定することはできません。

インターフェイスのメソッドは宣言できますが、定義できません。

次の例では、VJS1254 エラーが生成されます。

```
// VJS1254.jsl
interface MyInterface
{
   void Test() {}   // VJS1254
   // try the following line instead
   // void Test();
}

public class MyClass1
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1255

エラー メッセージ
抽象メソッド '<メソッド名>' に本体を指定することはできません。

abstract としてマークされたメソッドは宣言できますが、定義できません。

次の例では、VJS1255 エラーが生成されます。

```
// VJS1255.jsl
abstract public class MyClass
{
   public abstract void Test(){}   // VJS1255
   // try the following line instead
   //  public abstract void Test();
   // or delete the abstract keyword and keep the method body
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1256

エラー メッセージ
ネイティブ メソッド '<メソッド名>' に本体を指定することはできません。

native としてマークされたメソッドは宣言できますが、定義できません。

次の例では、VJS1256 エラーが生成されます。

```
// VJS1256.jsl
//
public class MyClass
{
   public native void Test(){}   // VJS1256
   // try the following line instead
   // public native void Test();
   // or delete the native keyword and keep the method body
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1257

エラー メッセージ
メソッド '<メソッド名>' は抽象ではなく、本体も定義されていません。

クラスのメソッドが定義されていません。

次の例では、VJS1257 エラーが生成されます。

```
// VJS1257.jsl
//
public class MyClass
{
   public void Test();   // VJS1257
   // try the following line instead
   // public void Test(){}
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1259

エラー メッセージ
メソッド '<メソッド名>' は値を返さなければなりません。

メソッドのシグネチャには戻り値が指定されていますが、メソッド実装の本体で **return** ステートメントが見つかりませんでした。

次の例では、VJS1259 エラーが生成されます。

```
// VJS1259.jsl
public class MyClass
{
   public int Test()   // VJS1259, uncomment return statement to resolve
   {
      // return 0;
   }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1265

エラー メッセージ
return ステートメントは初期化子では使用できません。

初期化子ブロックで **return** ステートメントが見つかりました。

次の例では、VJS1265 エラーが生成されます。

```
// VJS1265.jsl
public class MyClass   // VJS1265
{
   {
      return;   // return not allowed in initializer block
   }
   public static void main (String [] args)
   {
   }
}
```

コンパイラ エラー VJS1271

エラー メッセージ
アクセス修飾子は 1 つだけ使用できます。

アクセス修飾子が 2 回以上指定されています。

次の例では、VJS1271 エラーが生成されます。

```
// VJS1271.jsl
public class MyClass
{
   public private void Test()   // VJS1271, delete one public
   {
   }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1275

エラー メッセージ
クラス名 '<名前>' がパッケージ名と競合しています。

モジュールのクラス名と別のモジュールのパッケージ名が同じです。

使用例

```
// VJS1275a.jsl
// compile with: /target:library
package Test;
public class MyClass {}
```

次の例では、VJS1275 エラーが生成されます。

```
// VJS1275b.jsl
// compile with: VJS1275a.jsl /target:library
public class Test {}   // VJS1275
```

コンパイラ エラー VJS1276

エラー メッセージ
continue ステートメントで使用するラベル '<ラベル名>' は loop ステートメントをポイントしなければなりません。

continue ステートメントのラベルがループを指していませんでした。

次の例では、VJS1276 エラーが生成されます。

```
// VJS1276.jsl
public class MyClass
{
   public static void main()
   {
      label1: for (int i = 0; i < 100; ++i)
      {
         label2:
         {
            continue label2;   // VJS1276
            // try the following line instead
            // continue label1;
         }
      }
   }
}
```

コンパイラ エラー VJS1277

エラー メッセージ
'<メンバ名>' がサポートされていません。

メンバはサポートされていないので、アクセスできません。

使用例

```
// VJS1277a.cs
// compile with: /target:library
// a C# program
public class MyClass
{
   [System.ObsoleteAttribute("", true)]
   public static void m() {}
}
```

次の例では、VJS1277 エラーが生成されます。

```
// VJS1277.jsl
// compile with: /reference:vjs1277a.dll
public class MyClass2
{
   public static void main(String[] args)
   {
      MyClass.m();   // VJS1277
   }
}
```

コンパイラ エラー VJS1278

エラー メッセージ
同じクラス内で、メソッド '<メソッド 1>' と '<メソッド 2>' を定義することはできません。

次のメソッドの組み合わせは、同じクラスで使用できません。

- `String toString()` および `String ToString()`
- `Boolean equals(Object)` および `Boolean Equals(Object)`

次の例では、VJS1278 エラーが生成されます。

```
// VJS1278.jsl
public class MyClass   // VJS1278
{
   public String toString() {}
   public String ToString() {}
   public static void main(String[] args)
   {
      new MyClass() { };
   }
}
```

コンパイラ エラー VJS1279

エラー メッセージ
'<メンバ名>' は、匿名クラスによって実装されなければなりません。

匿名クラスは抽象でないため、このクラスに含まれるすべての抽象を実装する必要があります。

次の例では、VJS1279 エラーが生成されます。

```
// VJS1279.jsl
public abstract class MyClass
{
   abstract void m();
   public static void main(String[] args)
   {
      new MyClass() { };   // VJS1279
   }
}
```

コンパイラ エラー VJS1280

エラー メッセージ
'<型>' は、入れ子にされたクラス、またはクラス '<クラス名>' のインターフェイスではありません。

クラス内で、入れ子になったクラスまたはインターフェイス以外の型をインスタンス化しようとしました。

次の例では、VJS1280 エラーが生成されます。

```
// VJS1280.jsl
public class MyClass
{
   public static void main(String[] args)
   {
      new MyClass().new Inner();   // VJS1280
   }
}
```

コンパイラ エラー VJS1281

エラー メッセージ
あいまいなメンバ : 継承された '<メンバ 1>' および外部スコープ '<メンバ 2>' です。明示的な 'this' 修飾子が必要です。

コンパイラは、アクセスしようとしたメンバを見つけることはできませんでした。

次の例では、VJS1281 エラーが生成されます。

```
// VJS1281.jsl
// compile with: /target:library
class Base {
    int i;
}

class Outer {
    int i;
    class Derived extends Base {
        int j = i;
    }
}
```

コンパイラ エラー VJS1282

エラー メッセージ
型 '<型>' を宣言できません。

System.Object 型と **System.String** 型は、ソースファイルで宣言できません。

次の例では、VJS1282 エラーが生成されます。

```
// VJS1282.jsl
// compile with: /target:library /nowarn:1513
package System;   // VJS1282
class Object {
}
```

コンパイラ エラー VJS1284

エラー メッセージ
属性、値型または列挙型として内部クラスを指定できません

このエラーは、内部クラスが、属性、値型、または列挙型として宣言されている場合に発生します。

このエラーを解決するには

- 属性、値型、または列挙型として宣言されている内部クラスを特定します。

使用例

次の例では、VJS1284 エラーが生成されます。

```
public class MyClass
{
    public final class Point extends System.ValueType
    {
        public int x;
        public int y;
    }
}
```

参照

その他の技術情報

Visual J# コンパイラ

コンパイラ エラー VJS1300

エラー メッセージ
'<値>' は '<型>' の範囲にありません。

型に代入された値は、サポートされている範囲内の値ではありませんでした。

次の例では、VJS1301 エラーが生成されます。

```
// VJS1300.jsl
class MyClass
{
   public static void main (String [] args)
   {
      byte i = 256;   // VJS1300, 256 out of range for byte
      // try the following line instead
      // byte i = 127;
   }
}
```

コンパイラ エラー VJS1301

エラー メッセージ
型 '<型 1>' を '<型 2>' に割り当てることはできません。

無効な代入を試みました。

次の例では、VJS1301 エラーが生成されます。

```
// VJS1301.jsl
public class MyClass
{
   public static void main()
   {
      int i = 2.2;   // VJS1301
      // try the following lines instead
      // int i = 2;
      // i++;
   }
}
```

コンパイラ エラー VJS1302

エラー メッセージ
静的コンテキストから、静的でないフィールド '<フィールド名>' にアクセスすることはできません。

インスタンス フィールドをスタティック フィールドと見なしてアクセスを試みました。

次の例では、VJS1302 エラーが生成されます。

```
// VJS1302.jsl
// compile with: /target:library
class MyClass
{
   public void MyMethod()
   {
      Test.x = 10;   // VJS1302 x is not static in class Test
      int local = new Test ().x;   // OK, accessing instance field
      Test.y = 20;   // OK, accessing static field
   }
}

class Test
{
   public int x;   // instance field
   public static int y;   // static field
}
```

コンパイラ エラー VJS1303

エラー メッセージ
静的コンテキストから、静的でないメソッド '<メソッド名>' にアクセスすることはできません。

インスタンス メソッドにアクセスするには、外側の型のインスタンスを使う必要があります。

次の例では、VJS1303 エラーが生成されます。

```
// VJS1303.jsl
class MyClass
{
   public void Test()
   {
   }
   public static void main (String [] args)
   {
      MyClass.Test();   // VJS1303
      // try the following lines instead
      // MyClass x = new MyClass();
      // x.Test();
   }
}
```

コンパイラ エラー VJS1304

エラー メッセージ
クラス '<クラス名>' のコンストラクタにアクセスできません。

コンストラクタにアクセスできませんでした。

次の例では、VJS1304 エラーが生成されます。

```
// VJS1304.jsl
class MyClass
{
   public static void main (String [] args)
   {
      new Class1();   // VJS1304  constructor is private
      new Class1(10);   // OK, constructor is public
   }
}

class Class1
{
   private Class1()
   {
   }
   public Class1(int x)
   {
   }
}
```

コンパイラ エラー VJS1305

エラー メッセージ
条件式の条件は boolean でなければなりません。

条件式で評価される式は、ブール式であることが必要です。

次の例では、VJS1305 エラーが生成されます。

```
// VJS1305.jsl
class Test
{
   public static void main(String[] args)
   {
      int i = 0;
      boolean j = (i) ? true : false;   // VJS1305
      // try the following line instead
      // boolean j = (i == 0) ? true : false;
   }
}
```

コンパイラ エラー VJS1306

エラー メッセージ
型 '<型 1>' および '<型 2>' は条件式に対して互換性がありません。

条件式に互換性のない型が含まれています。

次の例では、VJS1306 エラーが生成されます。

```
// VJS1306.jsl
class MyClass
{
   public static void main (String [] args)
   {
      int j = 3 < 10 ? 1 : new Object();   // VJS1306 object not int
      int k = 3 < 10 ? 1 : 3;   // OK
   }
}
```

コンパイラ エラー VJS1307

エラー メッセージ
型 '<型>' は抽象クラスであるため、この型の新しいオブジェクトを作成することはできません。

抽象クラスはインスタンス化できません。

次の例では、VJS1307 エラーが生成されます。

```
// VJS1307.jsl
abstract public class MyClass1
{
   public abstract void Test();
}

public class MyClass2
{
   public static void main()
   {
      MyClass1 x = new MyClass1();   // VJS1307
   }
}
```

コンパイラ エラー VJS1308

エラー メッセージ
型 '<型>' はインターフェイスであるため、この型の新しいオジェクトを作成することはできません。

オブジェクトは、インターフェイスでインスタンス化できません。

次の例では、VJS1308 エラーが生成されます。

```
// VJS1308.jsl
class MyClass
{
   public static void main (String [] args)
   {
      MyInterface i = new MyInterface();   // VJS1308
   }
}

interface MyInterface
{
}
```

コンパイラ エラー VJS1309

エラー メッセージ
'new' を使用して、プリミティブ型のインスタンスを作成することはできません。

new 演算子を使ってプリミティブ型のインスタンス オブジェクトを作成することはできません。

次の例では、VJS1309 エラーが生成されます。

```
// VJS1309.jsl
class Test
{
   public static void main(String[] args)
   {
      int i = new int();   // VJS1309
      // try the following line instead
      // int i;
   }
}
```

コンパイラ エラー VJS1312

エラー メッセージ
'<型>' に、後置形式でない演算子を適用することはできません。

後置インクリメント演算子 (++) が変数に適用されましたが、変数の型では後置演算子がサポートされていません。

次の例では、VJS1312 エラーが生成されます。

```
// VJS1312.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      x++;   // VJS1312
   }
}
```

コンパイラ エラー VJS1313

エラー メッセージ
割り当て不可能な項目に、後置形式の演算子を適用することはできません。

後置演算子が、実行できないトークンに適用されました。

次の例では、VJS1313 エラーが生成されます。

```
// VJS1313.jsl
class Test
{
   public static void main(String[] args)
   {
      int i = 0;
      0++;   // VJS1313
      // try the following line instead
      // i++;
   }
}
```

コンパイラ エラー VJS1314

エラー メッセージ
初期化されていない変数に、後置形式の演算子を適用することはできません。

後置インクリメント演算子 (++) が、初期化されていない可能性のある変数に適用されました。

次の例では、VJS1314 エラーが生成されます。

```
// VJS1314.jsl
class Test
{
   public static void main(String[] args)
   {
      int i;
      i++;   // VJS1314, initialize i first
   }
}
```

コンパイラ エラー VJS1315

エラー メッセージ
'<型>' にプレフィックス演算子を適用することはできません。

プレフィックス インクリメント演算子 (++) が変数に適用されましたが、変数の型ではプレフィックス演算子がサポートされていません。

次の例では、VJS1315 エラーが生成されます。

```
// VJS1315.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      --x;   // VJS1315
   }
}
```

コンパイラ エラー VJS1316

エラー メッセージ
割り当て不可能な項目にプレフィックス演算子を適用することはできません。

プレフィックス演算子が実行できないトークンに適用されました。

次の例では、VJS1316 エラーが生成されます。

```
// VJS1316.jsl
class Test
{
   public static void main(String[] args)
   {
      int i = 0;
      --0;   // VJS1316
      // try the following line instead
      // --i;
   }
}
```

コンパイラ エラー VJS1317

エラー メッセージ
初期化されていない変数に、プレフィックス演算子を適用することはできません。

前置演算子が初期化されていない可能性のある変数に適用されました。

次の例では、VJS1317 エラーが生成されます。

```
// VJS1317.jsl
class Test
{
   public static void main(String[] args)
   {
      int i;
      --i;   // VJS1317, initialize i first
   }
}
```

コンパイラ エラー VJS1318

エラー メッセージ
単項演算子 + / - を '<型>' に適用することはできません。

単項演算子を使用できるのは数値型だけです。

次の例では、VJS1318 エラーが生成されます。

```
// VJS1318.jsl
class MyClass
{
   public static void main (String [] args)
   {
      Object obj =- (new Object());   // VJS1318
   }
}
```

コンパイラ エラー VJS1319

エラー メッセージ
単項演算子 ~ を '<オペランド名>' に適用することはできません。

単項演算子 ~ は、このオペランドでは使用できません。

次の例では、VJS1319 エラーが生成されます。

```
// VJS1319.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      x = ~x;   // VJS1319
      // try the following lines instead
      // byte i = 25;
      // byte j = ~25;
   }
}
```

コンパイラ エラー VJS1320

エラー メッセージ
単項演算子 ! を '<型>' に適用することはできません。

単項演算子 (!) が、否定をサポートしていない型に適用されました。

次の例では、VJS1320 エラーが生成されます。

```
// VJS1320.jsl
class Test
{
   public static void main(String[] args)
   {
      int i;
      boolean j = true;
      i = 0;
      if (!i)   // VJS1320
      // try the following line instead
      // if (!j)
      {
      }
   }
}
```

コンパイラ エラー VJS1321

エラー メッセージ
'==' 演算子を '<型 1>' および '<型 2>' に適用することはできません。

比較のサポート対象でない型の間で、比較を試みました。

次の例では、VJS1321 エラーが生成されます。

```
// VJS1321.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x == 1)   // VJS1321
      {
      }
   }
}
```

コンパイラ エラー VJS1322

エラー メッセージ
関係演算子を 'type1' および 'type2' に適用することはできません。

関係演算子のサポート対象でない型の間で、関係演算を試みました。

次の例では、VJS1322 エラーが生成されます。

```
// VJS1322.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x < 1)   // VJS1322
      {
      }
   }
}
```

コンパイラ エラー VJS1323

エラー メッセージ
'+' 演算子を '<型 1>' および '<型 2>' に適用することはできません。

加算のサポート対象でない型の間で加算を試みました。

次の例では、VJS1323 エラーが生成されます。

```
// VJS1323.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x + 1)   // VJS1323
      {
      }
   }
}
```

コンパイラ エラー VJS1324

エラー メッセージ
'*' 演算子を '<型 1>' および '<型 2>' に適用することはできません。

乗算のサポート対象でない型の間で乗算を試みました。

次の例では、VJS1324 エラーが生成されます。

```
// VJS1324.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x * 1)   // VJS1324
      {
      }
   }
}
```

コンパイラ エラー VJS1325

エラー メッセージ
'/' 演算子を '<型 1>' および '<型 2>' に適用することはできません。

除算のサポート対象でない型の間で除算を試みました。

次の例では、VJS1325 エラーが生成されます。

```
// VJS1325.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x / 1)   // VJS1325
      {
      }
   }
}
```

コンパイラ エラー VJS1326

エラー メッセージ
'-' 演算子を '<型 1>' および '<型 2>' に適用することはできません。

減算のサポート対象でない型の間で減算を試みました。

次の例では、VJS1326 エラーが生成されます。

```
// VJS1326.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x - 1)   // VJS1326
      {
      }
   }
}
```

コンパイラ エラー VJS1327

エラー メッセージ
'%' 演算子を '<型 1>' および '<型 2>' に適用することはできません。

剰余演算のサポート対象でない型の間で剰余演算を試みました。

次の例では、VJS1327 エラーが生成されます。

```
// VJS1327.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x % 1)   // VJS1327
      {
      }
   }
}
```

コンパイラ エラー VJS1328

エラー メッセージ
ビットごとの &、^、| 演算子を 'type1' および 'type2' に適用することはできません。

操作のサポート対象でない型の間で操作を実行しようとしました。

次の例では、VJS1328 エラーが生成されます。

```
// VJS1328.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x & 1)   // VJS1328
      {
      }
   }
}
```

コンパイラ エラー VJS1329

エラー メッセージ
シフト演算子 <<、>>、>>> を 'type1' および 'type2' に適用することはできません。

操作のサポート対象でない型の間で操作を実行しようとしました。

次の例では、VJS1329 エラーが生成されます。

```
// VJS1329.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x << 1)   // VJS1329
      {
      }
   }
}
```

コンパイラ エラー VJS1330

エラー メッセージ
数値を 0 で割ることはできません。

0 による除算は定義されていないため、無効な演算です。

次の例では、VJS1330 エラーが生成されます。

```
// VJS1330.jsl
class Test
{
   public static void main(String[] args)
   {
      int i = 1/0;   // VJS1330
   }
}
```

コンパイラ エラー VJS1331

エラー メッセージ
'&&' 演算子を '<型 1>' および '<型 2>' に適用することはできません。

操作のサポート対象でない型の間で操作を実行しようとしました。

次の例では、VJS1331 エラーが生成されます。

```
// VJS1331.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x && 1)   // VJS1331
      {
      }
   }
}
```

コンパイラ エラー VJS1332

エラー メッセージ
'||' 演算子を '<型 1>' および '<型 2>' に適用することはできません。

操作のサポート対象でない型の間で操作を実行しようとしました。

次の例では、VJS1332 エラーが生成されます。

```
// VJS1332.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x || 1)   // VJS1332
      {
      }
   }
}
```

コンパイラ エラー VJS1333

エラー メッセージ
'<型 1>' から '<型 2>' にはキャストできません。

キャストのサポート対象でない型の間でキャストを試みました。

次の例では、VJS1333 エラーが生成されます。

```
// VJS1333.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      int i = 0;
      i = (int)x;   // VJS1333
   }
}
```

コンパイラ エラー VJS1334

エラー メッセージ
型 '<型 1>' を '<型 2>' として返すことはできません。

メソッドが予測とは異なる型を返しました。

次の例では、VJS1334 エラーが生成されます。

```
// VJS1334.jsl
class Test
{
   public int Test()
   {
      Test x = new Test();
      return x;   // VJS1334
      // try the following line instead
      // return 0;
   }
   public static void main(String[] args)
   {
   }
}
```

コンパイラ エラー VJS1335

エラー メッセージ
void でないメソッドからは値を返さなければなりません。

メソッドの戻り値の型は void ではありませんでしたが、メソッドの本体には値なしの **return** ステートメントがあります。

次の例では、VJS1335 エラーが生成されます。

```
// VJS1335.jsl
// compile with: /target:library
public class MyClass
{
   public int Test()
   {
      return;   // VJS1335
      // try the following line instead
      // return 0;
   }
}
```

コンパイラ エラー VJS1336

エラー メッセージ
void メソッドから値を返すことはできません。

戻り値の型が void としてマークされているメソッドから、値を返すことはできません。

次の例では、VJS1336 エラーが生成されます。

```
// VJS1336.jsl
public class MyClass
{
   public static void Test()
   {
      return 0;   // VJS1336, delete return value
   }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1337

エラー メッセージ
コンストラクタから値を返すことはできません。

コンストラクタに **return** ステートメントを含めることはできません。

次の例では、VJS1337 エラーが生成されます。

```
// VJS1337.jsl
class Test
{
   Test()
   {
      return 0;   // VJS1337, delete return statement
   }
   public static void main(String[] args)
   {
   }
}
```

コンパイラ エラー VJS1338

エラー メッセージ
演算子 'instanceof' を 'type1' および 'type2' に適用することはできません。

instanceof 演算に、互換性のない 2 つの型が含まれていました。

次の例では、VJS1338 エラーが生成されます。

```
// VJS1338.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x instanceof Test2)   // VJS1338
      // try the following line instead
      // if (x instanceof Test)
      {
      }
   }
}

class Test2
{
}
```

コンパイラ エラー VJS1339

エラー メッセージ
'instanceof' の右側には、'<型>' ではなく参照型を指定しなければなりません。

instanceof 演算子の右側の識別子には、参照型を使う必要があります。

次の例では、VJS1339 エラーが生成されます。

```
// VJS1339.jsl
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x instanceof int)   // VJS1339
      // try the following line instead
      // if (x instanceof Test)
      {
      }
   }
}
```

コンパイラ エラー VJS1340

エラー メッセージ
配列インデックスがありません。

配列操作に必要な配列のインデックスがありません。

次の例では、VJS1340 エラーが生成されます。

```
// VJS1340.jsl
class Test
{
   public static void main(String[] args)
   {
      int[] x = new int[5];
      x[] = 0;   // VJS1340
      // try the following line instead
      // x[0] = 0;
   }
}
```

コンパイラ エラー VJS1341

エラー メッセージ
配列バウンドは初期化子で指定できません。

初期化リストを使って配列を初期化する場合は、配列の次元を指定できません。

次の例では、VJS1341 エラーが生成されます。

```
// VJS1341.jsl
// compile with: /target:library
class MyClass
{
   public void Test()
   {
      int arr[] = new int[2]{1,2}; // VJS1341
      // try the following line instead
      // int arr[] = new int[]{1,2};
      arr[0] = 0;
   }
}
```

コンパイラ エラー VJS1342

エラー メッセージ
配列アクセサは序数でなければなりませんが、'<型>' が指定されました。

配列の要素にアクセスするときに、整数型でないインデックスが使用されました。

次の例では、VJS1342 エラーが生成されます。

```
// VJS1342.jsl
public class MyClass
{
   public static void main()
   {
      int arr[] = {1,2};
      String str = "test";
      arr[str] = 1;   // VJS1342, str is type String, not int
      // try the following line instead
      arr[0] = 1;
   }
}
```

コンパイラ エラー VJS1343

エラー メッセージ
配列アクセサのプレフィックスは配列型でなければなりませんが、'<型>' が指定されました。

配列アクセサ演算子 ([]) が、配列でない変数に使用されました。

次の例では、VJS1343 エラーが生成されます。

```
// VJS1343.jsl
public class MyClass
{
   public static void main()
   {
      int i = 0;
      int arr[] = {1,2};
      i[0] = 1;   // VJS1343 i is not an array so i[0] is not valid
      // try the following line instead
      arr[0] = 1;
   }
}
```

コンパイラ エラー VJS1344

エラー メッセージ
配列バウンドは序数でなければなりませんが、'<型>' が指定されました。

new 演算子の呼び出しなどで配列サイズを指定する場合、変数の型は序数であることが必要です。

次の例では、VJS1344 エラーが生成されます。

```
// VJS1344.jsl
public class MyClass
{
   public static void main()
   {
      String s = "test";
      int i = 10;
      int arr[] = new int[s]; // VJS1344, s is String not ordinal
      // try the following line instead
      int arr2[] = new int[i];
   }
}
```

コンパイラ エラー VJS1345

エラー メッセージ
値の型 '<型>' が必要ですが、配列初期化子が見つかりました。

非配列型を配列初期化子で初期化しようとしました。

次の例では、VJS1345 エラーが生成されます。

```
// VJS1345.jsl
public class MyClass
{
   int i;
   public static void main()
   {
      MyClass c = {1,2};   // VJS1345
      // try the following line instead
      MyClass c1 = new MyClass();
      c1.i = 0;
   }
}
```

コンパイラ エラー VJS1346

エラー メッセージ
配列の作成では、少なくとも 1 つの次元を指定しなければなりません。

配列宣言の形式が不正です。配列の次元が指定されていません。

次の例では、VJS1346 エラーが生成されます。

```
// VJS1346.jsl
class Test
{
   public static void main(String[] args)
   {
      int[] x = new int[];   // VJS1346
      // try the following line instead
      // int[] x = new int[5];
   }
}
```

コンパイラ エラー VJS1347

エラー メッセージ
四角形の配列の範囲は、指定があれば、すべて指定しなければなりません。

四角形配列の宣言の形式が不正です。

次の例では、VJS1347 エラーが生成されます。

```
// VJS1347.jsl
class MyClass
{
   public static void main()
   {
   int[,] arr = new int[10,];   // VJS1347, a dimension is missing.
   int[,] arr2 = new int[10,10];   // OK, all dimensions are present
   }
}
```

コンパイラ エラー VJS1348

エラー メッセージ
配列の作成で、指定されていない次元の後に指定された次元を使用することはできません。

配列の最初の次元が指定されずに 2 番目の次元が指定されていますが、このように指定することは許可されていません。

次の例では、VJS1348 エラーが生成されます。

```
// VJS1348.jsl
public class MyClass
{
   public static void main()
   {
      int[][] arr = new int[][2];   // VJS1348
      // try the following line instead
      int[][] arr2 = new int[3][2];
   }
}
```

コンパイラ エラー VJS1350

エラー メッセージ
不適切な初期化子が '<宣言>' に適用されました。

指定された初期化子は四角形配列に対応していません。

次の例では、VJS1350 エラーが生成されます。

```
// VJS1350.jsl
class MyClass
{
   public static void main()
   {
      int[,] arr = new int[,]{{1,2},{3,4,5}};   // VJS1350
      // Rectangular arrays should have all rows of the same size
      // try the following line instead
      int[,] arr2 = new int[,]{{1,2},{3,4}};
      arr2[0,0] = 0;
   }
}
```

コンパイラ エラー VJS1351

エラー メッセージ
不適切なインデックスの数字が '<宣言>' に適用されました。

四角形配列の要素へのアクセスを試みたときに、指定されたインデックス数と配列の次元数が一致しませんでした。

次の例では、VJS1351 エラーが生成されます。

```
// VJS1351.jsl
class MyClass
{
   public static void main()
   {
      int[,] arr = new int[2,2];   // 2x2 rectangular array
      arr[0] = 1;   // VJS1351, must specify two indices
      // try the following line instead
      arr[0,1] = 10;
   }
}
```

コンパイラ エラー VJS1354

エラー メッセージ
静的コンテキストで 'this' は使用できません。

this 演算子はスタティック メソッドで使用できません。

次の例では、VJS1354 エラーが生成されます。

```
// VJS1354.jsl
class Test
{
   public static void Test(Test t)
   {
      if (this == t)   // VJS1354
      {
      }
   }
   public static void main(String[] args)
   {
   }
}
```

コンパイラ エラー VJS1355

エラー メッセージ
現在のコンテキストでコンストラクタの呼び出しに 'this' は使用できません。

this 構文を使ってクラス コンストラクタを呼び出すときは、クラスのインスタンス変数を使用できません。インスタンス変数を使用できないのは、**this** 変数がまだ初期化されていないためです。

次の例では、VJS1355 エラーが生成されます。

```
// VJS1355.jsl
// compile with: /target:library
public class MyClass
{
   int i;   // an instance variable
   static final int ONE = 1;   // a static variable
   public MyClass()
   {
      this(i);   // VJS1355
      // try the following line instead
      // this(ONE); // OK. ONE is a static variable
   }
   public MyClass(int i)
   {
   }
}
```

コンパイラ エラー VJS1356

エラー メッセージ
'.this' へのプレフィックスは、'<型>' ではなく、クラス名でなければなりません。

this の呼び出しの形式が不正です。

次の例では、VJS1356 エラーが生成されます。

```
// VJS1356.jsl
// compile with: /target:library
public class OuterClass
{
   public int i;
   public class InnerClass
   {
      void Test()
      {
         int.this.i = 20;   // VJS1356, prefix int is not a class name
         // try the following line instead
         // OuterClass.this.i = 20; // OK
      }
   }
}
```

コンパイラ エラー VJS1357

エラー メッセージ
クラス '<クラス名>' は外側のクラスではありません。

修飾された **this** ステートメントを使って外側のクラスの **this** 変数にアクセスしようとしましたが、**this** 式のプリフィックスは外側のクラスではありません。

次の例では、VJS1357 エラーが生成されます。

```
// VJS1357.jsl
// compile with: /target:library
public class OuterClass
{
   public int i;
   public class InnerClass
   {
      void Test()
      {
         InnerClass2.this.j = 10;   // VJS1357, not an enclosing class
         // try the following line instead
         // OuterClass.this.i = 10;
      }
   }

   public class InnerClass2
   {
      public int j;
   }
}
```

コンパイラ エラー VJS1358

エラー メッセージ
catch または finally ブロックが必要です。

不完全な **try** ステートメントが検出されました。

このエラーを解決するには

1. **try** ステートメントを探します。
2. ステートメント全体を見渡し、最後に **catch** ブロックまたは **finally** ブロックが記述されていることを確認します。

使用例
次の例では、VJS1358 エラーが生成されます。

```
// VJS1358.jsl
// compile with: /target:library
// VJS1358 expected

class MyClass
{
   void Test()
   {
      try {}
   }

   void Test2()
   {
      try {}
      catch (Exception e) {}
   }
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1359

エラー メッセージ
java.lang.Throwable、または System.Exception から継承しない '<クラス名>' はキャッチできません。

catch 句に指定された例外は、例外クラスから派生していません。

次の例では、VJS1359 エラーが生成されます。

```
// VJS1359.jsl
class MyClass
{
}

class MyClass2
{
   public static void main(String[] args)
   {
      try
      {
      }
      catch (MyClass e)   // VJS1359, MyClass is not an exception class
      // try the following line instead
      // catch (Exception e)
      {
      }
   }
}
```

コンパイラ エラー VJS1360

エラー メッセージ
到達できない catch: 対応する例外が try ブロックでスローされていません。

try ブロックでスローされた例外クラスまたは派生した例外がないため、**catch** 句は到達できません。

次の例では、VJS1360 エラーが生成されます。

```
// VJS1360.jsl
class MyClass extends Exception
{
}

class MyClass2
{
   public static void main(String[] args)
   {
      try
      {
      // uncomment the following line to resolve:
      //    throw new MyClass();
      }
      catch (MyClass e)   // VJS1360, Exception MyClass is not thrown in the try block
      {
      }
      catch (Exception e)
      {
      }
   }
}
```

コンパイラ エラー VJS1361

エラー メッセージ
到達できない Catch: 対応する例外はこれより前の catch 句でキャッチされました。

catch 句が重複しています。

次の例では、VJS1361 エラーが生成されます。

```
// VJS1361.jsl
class MyClass
{
}

class MyClass2
{
   public static void main(String[] args)
   {
      try
      {
      }
      catch (Exception e)
      {
      }

      catch (Exception e)   // VJS1361, duplicate catch block
      {
      }
   }
}
```

コンパイラ エラー VJS1362

エラー メッセージ
java.lang.Throwable、または System.Exception から継承しない '<クラス名>' はスローできません。

throw ステートメントでスローしたクラスは、例外クラスではありません。

次の例では、VJS1362 エラーが生成されます。

```
// VJS1362.jsl
class MyClass
{
}

class MyClass2 extends Exception
{
}

class MyClass3
{
   public static void main(String[] args)
   {
      try
      {
         throw new MyClass();   // VJS1362, MyClass is not exception class
        // try the following line instead
        // throw new MyClass2();
      }

       catch (MyClass2 e)
       {
       }
   }
}
```

コンパイラ エラー VJS1363

エラー メッセージ
型 '<クラス名>' にアクセスできません。

クラス (**class**) は、パッケージ スコープ クラスです。別のモジュールのクラスが、他のパッケージにあるパッケージ スコープ クラスの拡張を試みています。別のクラスのパッケージ スコープ メンバにアクセスできるのは、アクセスするクラスが同じパッケージにある場合だけです。

使用例

```
// VJS1363a.jsl
// compile with: /target:library
package p;
class Base {}
public class Base2 {}

class MyClass extends p.Base {}   // OK
```

次の例では、VJS1363 エラーが生成されます。

```
// VJS1363b.jsl
// compile with: /target:library VJS1363a.jsl
class MyClass2 extends p.Base {}   // VJS1363
class MyClass3 extends p.Base2 {}   // OK
```

コンパイラ エラー VJS1364

エラー メッセージ
フィールド '<フィールド名>' のプレフィックスは、'<型>' ではなく、参照でなければなりません。

インスタンス フィールドにアクセスするには、型のインスタンスを使う必要があります。

このエラーは、**int**、**char**、**boolean** などのプリミティブ型のフィールドにアクセスを試みたことを示します。プリミティブ型は、フィールドを持つことができません。フィールドやメソッドを持つことができるのは、参照型だけです。

次の例では、VJS1364 エラーが生成されます。

```
// VJS1364.jsl
class MyClass
{
   int i;
   public void Test()
   {
      MyClass x = new MyClass();
      i.i = 0;   // VJS1364, i is not a reference to the class
      // try the following line instead
      // x.i = 0;
   }
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1365

エラー メッセージ
フィールド '<フィールド名>' がクラス '<クラス名>' で見つかりません。

宣言されていないフィールドへの参照がクラスで見つかりました。

Visual J# では、プロパティ アクセサ メソッドに特別な重要性が追加されることはありません。プロパティ アクセサ メソッドは、クラスのメソッドとして扱われます。

使用例

```
// VJS1365.jsl
// compile with: /target:library
class Test
{
   int i;
   public void Test(Test t)
   {
      this.t = 0;   // VJS1365
      this.i = 0;   // OK
   }
}
```

次の例では、VJS1365 エラーが生成されます。

```
// VJS1365b.jsl
public class MyClass
{
   /** @property */
   void set_Val(int i)
   {
       value = i;
   }

   public static void main(String [] args)
   {
       MyClass c = new MyClass();
       c.Val = 10;   // VJS1365
       c.set_Val(10);   // OK
   }

   private int value;
}
```

コンパイラ エラー VJS1366

エラー メッセージ
フィールド '<フィールド名>' にアクセスできません。

フィールドにアクセスできませんでした。

次の例では、VJS1366 エラーが生成されます。

```
// VJS1366.jsl
class MyClass
{
   private int i;
}

class MyClass2
{
   public static void main()
   {
      MyClass x = new MyClass();
      x.i = 0;   // VJS1366, i is private
   }
}
```

コンパイラ エラー VJS1367

エラー メッセージ
このオブジェクトを通して、継承フィールド '<フィールド名>' にアクセスできません。

継承フィールドにアクセスできませんでした。基本クラスのパッケージに派生クラスを追加してください。

使用例

```
// VJS1367a.jsl
// compile with: /target:library
package p;

public class Base
{
   protected int f;
}

public class Derived extends Base {}
```

次の例では、VJS1367 エラーが生成されます。

```
// VJS1367b.jsl
// compile with: VJS1367a.jsl
// add MyClass to package p to resolve
class MyClass
{
   public static void main (String [] args)
   {
      int k = new p.Derived().f;   // VJS1367
   }
}
```

コンパイラ エラー VJS1368

エラー メッセージ
静的な内部クラスから、静的でないフィールド '<フィールド名>' にアクセスできません。

スタティック内部クラスのクラス スコープから外部クラスのインスタンス メンバにはアクセスできません。

次の例では、VJS1368 エラーが生成されます。

```
// VJS1368.jsl
// compile with: /target:library
class MyClass
{
   int f = 2;
   static int f2 = 10;
   static class Inner
   {
      int k = f;   // VJS1368, f is an instance variable
      int k2 = f2;   // OK: f2 is a static member
   }
   class Inner2
   {
      int k = f;   // OK: access instance member from instance inner class
   }
}
```

コンパイラ エラー VJS1369

エラー メッセージ
静的な内部クラスから、静的でないメソッド '<メソッド名>' にアクセスできません。

スタティック内部クラスから、スタティックでないメンバにはアクセスできません。

次の例では、VJS1369 エラーが生成されます。

```
// VJS1369.jsl
// compile with: /target:library
class MyClass
{
   int instanceMethod()
   {
      return 1;
   }
   static int staticMethod()
   {
      return 2;
   }
   static class Inner
   {
      int k = instanceMethod();   // VJS1369
      int k2 = staticMethod ();   // OK
   }

   class Inner2
   {
      // OK: accessing an instance member from an instance inner class
      int k = instanceMethod ();
   }
}
```

コンパイラ エラー VJS1371

エラー メッセージ
ここで完全に宣言されていないフィールド '<フィールド名>' にはアクセスできません。

フィールドは、静的初期化子で使う前に宣言しておく必要があります。

次の例では、VJS1371 エラーが生成されます。

```
// VJS1371.jsl
class MyClass
{
   static
   {
      i = 2;   // VJS1371, declare i before this static initializer
   }

   static int i;

   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1372

エラー メッセージ
インターフェイス フィールド '<フィールド名>' は初期化子を含まなければなりません。

インターフェイス フィールドが初期化されていません。

次の例では、VJS1372 エラーが生成されます。

```
// VJS1372.jsl
interface Point
{
   int i;   // VJS1372
   // try the following line instead
   // int i = 0;
}

public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1373

エラー メッセージ
浮動小数点リテラル '<リテラル名>' に指数がありません。

浮動小数点値の指数部がありません。

次の例では、VJS1373 エラーが生成されます。

```
// VJS1373.jsl
class MyClass
{
   float f = 3.40282347e+f;   // VJS1373
   // try the following line instead
   // float f = 3.40282347e+38f;

   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1374

エラー メッセージ
'<リテラル名>' は、型 'float' のリテラルに対して有効な値ではありません。

不正な形式の浮動小数点値が検出されました。

次の例では、VJS1374 エラーが生成されます。

```
// VJS1374.jsl
class MyClass
{
   float f = 11111111111111111111111111111111111111e+2f;   // VJS1374
   // try the following line instead
   // float f = 3.140e+30f;

   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1377

エラー メッセージ
ローカル変数 '<変数名>' は内部クラスで最後に使用されなければなりません。

外側のクラスで **final** としてマークされていないローカル変数は、内部クラスからアクセスできません。

次の例では、VJS1377 エラーが生成されます。

```
// VJS1377.jsl
// compile with: /target:library
class MyClass
{
   void Test()
   {
      int k = 30;
      final int k2 = 20;
      class Inner
      {
         int p = k;   // VJS1377
         int p2 = k2;   // OK
      }
   }
}
```

コンパイラ エラー VJS1379

エラー メッセージ
super はスタティック コンテキストでは使用できません。

キーワード **super** が正しく使用されていません。

コンパイラ エラー VJS1380

エラー メッセージ
break ステートメントはここでは使用できません。

break ステートメントが、無効な場所で見つかりました。

次の例では、VJS1380 エラーが生成されます。

```
// VJS1380.jsl
public class MyClass
{
   public static void main()
   {
      break;   // VJS1380
   }
}
```

コンパイラ エラー VJS1381

エラー メッセージ
文字リテラルが無効です : <リテラル名>

不正な形式の文字リテラルが検出されました。

次の例では、VJS1381 エラーが生成されます。

```
// VJS1381.jsl
class MyClass
{
   public static void main()
   {
      char c = 'a ';   // VJS1381, invalid char
      // try the following line instead
      char c = 'a';
   }
}
```

コンパイラ エラー VJS1382

エラー メッセージ
文字リテラルに無効な改行があります。

文字リテラルでは改行を使用できません。

次の例では、VJS1382 エラーが生成されます。

```
// VJS1382.jsl
// compile with: /target:library
class MyClass
{
   char s = '
   f';   // VJS1382
   // try the following line instead
   // char s = 'f';
}
```

コンパイラ エラー VJS1383

エラー メッセージ
文字リテラルに閉じかっこがありません : <リテラル名>

不正な形式の文字リテラルが検出されました。

次の例では、VJS1383 エラーが生成されます。

```
// VJS1383.jsl
class MyClass
{
   public static void main()
   {
      char c = 'a;   // VJS1383
      // try the following line instead
      // char c = 'a';
   }
}
```

コンパイラ エラー VJS1384

エラー メッセージ
リテラル文字列に閉じかっこがありません : <リテラル名>

不正な形式のリテラル文字列が検出されました。

次の例では、VJS1384 エラーが生成されます。

```
// VJS1384.jsl
class MyClass
{
   public static void main()
   {
      String s = "test;   // VJS1384
      // try the following line instead
      // String s = "test";
   }
}
```

コンパイラ エラー VJS1385

エラー メッセージ
8 進数のエスケープ シーケンス '<値>' が無効です。値は、0377 以下でなければなりません。

使用された 8 進値は、有効な値の範囲外です。

次の例では、VJS1385 エラーが生成されます。

```
// VJS1385.jsl
// compile with: /target:library
class MyClass
{
   // Error: octal char sequence should be less than o equal to 377
   char s = '\456';   // VJS1385
   // try the following line instead
   // char c = '\375';
}
```

コンパイラ エラー VJS1386

エラー メッセージ
エスケープ シーケンス '<シーケンス名>' が無効です。

無効なエスケープ シーケンスが見つかりました。8 進値の場合、有効な数字は 0 ～ 7 です。

次の例では、VJS1386 エラーが生成されます。

```
// VJS1386.jsl
// compile with: /target:library
class MyClass
{
   char s = '\8';   // VJS1386
   char c = '\7';   // OK
}
```

コンパイラ エラー VJS1387

エラー メッセージ
Unicode エスケープ シーケンス 'シーケンス' が無効です。

無効な Unicode エスケープ シーケンスが見つかりました。有効な形式は、**\u**#### です。

次の例では、VJS1387 エラーが生成されます。

```
// VJS1387.jsl
// compile with: /target:library
class MyClass
{
   char c = '\ug000';   // VJS1387
   // try the following line instead
   char cc = '\u1000';
}
```

コンパイラの警告 (レベル 1) VJS1393

エラー メッセージ
アセンブリ属性の定義は、パッケージとインポート ステートメントの後のファイルの最初の部分においてのみ行われます。

ソース コード ファイルで、型定義の後にアセンブリ レベルの属性が指定されています。

次の例では、VJS1393 エラーが生成されます。

```
// VJS1393.jsl
// compile with: /W:1
public class MyClass
{
   public static void main()
   {
   }
}
/** @assembly System.Reflection.AssemblyTitle("") */   // VJS1393, move to top of file
```

コンパイラの警告 (レベル 1) VJS1395

エラー メッセージ
.NET 言語拡張機能が無効にされているときは、この機能は使用できません。

使用されている言語拡張機能は、/x:net コンパイラ オプションと矛盾します。このオプションは、共通言語ランタイムをサポートする言語拡張機能を無効にします。たとえば、属性は無視されます。

次の例では、VJS1395 エラーが生成されます。

```
// VJS1395.jsl
// compile with: /x:net /W:1
/** @assembly System.Reflection.AssemblyTitle("") */  // VJS1395
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1396

エラー メッセージ
属性 '<属性名>' は抽象クラスであるため、アタッチできません。

抽象クラスに基づく属性は使用できません。

使用例

```
// VJS1396a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public abstract class MyAttribute : System.Attribute
{
   public MyAttribute () {}
}
```

```
// VJS1396b.jsl
// compile with: /reference:VJS1396a.dll /target:library
/** @attribute MyAttribute() */   // VJS1396 MyAttribute is abstract
class MyClass {}
```

コンパイラ エラー VJS1397

エラー メッセージ
属性 '<属性名>' は 'System.Attribute' から継承していません。

適用された属性は正しく定義されていません。

使用例

```
// VJS1397a.cs
// compile with: /target:library
// a C# program
// [System.AttributeUsage(System.AttributeTargets.All)]
public abstract class MyAttribute // : System.Attribute
{
   public MyAttribute ()
   {
   }
}
```

```
// VJS1397b.jsl
// compile with: /reference:VJS1397a.dll
/** @attribute MyAttribute() */   // VJS1397, MyAttribute not attr. class
// uncomment the attribute support syntax in VJS1397a.cs to resolve
class MyClass
{
  public static void main (String [] args)
   {
   }
}
```

コンパイラ エラー VJS1398

エラー メッセージ
属性 '<属性名>' は定数ではないコンストラクタ パラメータを持っています。

属性コンストラクタのすべてのパラメータは、定数であることが必要です。属性コンストラクタのパラメータとして変数を渡すことはできません。

解決するには、共通言語仕様 (CLS: common language specification) に準拠させるために属性を使用したり属性コンストラクタを変更したりしないでください。

使用例

```
// VJS1398a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttribute:System.Attribute {
   public MyAttribute (MyType t) {}
}

public class MyType {}
```

```
// VJS1398b.jsl
// compile with: /target:library /reference:VJS1398a.dll
/** @attribute MyAttribute (new MyType ()) */   // VJS1398
public class MyClass {}
```

コンパイラ エラー VJS1399

エラー メッセージ
属性 '<属性名>' は、1 次元ではない配列コンストラクタ パラメータを持っています。

2 次元以上の配列は、属性パラメータとして渡すことはできませんが、共通言語仕様 (CLS: Common Language Specification) には準拠しています。

使用例

```
// VJS1399a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttribute : System.Attribute
{
   public MyAttribute (int [][] a) {}
}

[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttribute2 : System.Attribute
{
   public MyAttribute2 (int [] a) {}
}
```

次の例では、VJS1399 エラーが生成されます。

```
// VJS1399b.jsl
// compile with: /target:library /reference:VJS1399a.dll
/** @attribute MyAttribute (new int[][]{{1,2},{1,2,3}}) */   // VJS1399
class MyClass {}

// OK
/** @attribute MyAttribute2 (new int[]{8}) */
class MyClass2 {}
```

コンパイラ エラー VJS1401

エラー メッセージ
属性パラメータが不適切です。

Visual J# では、共通言語仕様 (CLS: Common Language Specification) 準拠の型を持つ属性パラメータを使う必要があります。

CLS の型と Java 言語の型の対応を次の表に示します。

CLS の型	Java 言語の型
System.Byte(byte)	ubyte
System.Char(char)	char
System.Boolean(bool)	boolean
System.Int16(short)	short
System.Int32(int)	int
System.Int64(long)	long
System.Single(float)	float
System.Double(double)	double

使用例

```
// VJS1401a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttribute:System.Attribute {
   public MyAttribute (sbyte t) {}
}
```

次の例では、VJS1401 エラーが生成されます。

解決するには、CLS 準拠型にするために属性を使用したり属性コンストラクタを変更したりしないでください。

```
// VJS1401b.jsl
// compile with: /target:library /reference:VJS1401a.dll
/** @attribute MyAttribute ((byte)10) */   // VJS1401
class MyClass {}
```

コンパイラ エラー VJS1402

エラー メッセージ
名前付きフィールドまたはパラメータ 'param' が、型 'type' に見つかりません。

名前付きパラメータが存在しません。

使用例

```
// VJS1402a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttribute : System.Attribute
{
   public MyAttribute () {}
   public int Version = 20;
}
```

次の例では、VJS1402 エラーが生成されます。

```
// VJS1402b.jsl
// compile with: /target:library /reference:VJS1402a.dll
/** @attribute MyAttribute (NonExistingField = 10) */   // VJS1402
// try the following line instead
// /** @attribute MyAttribute (Version = 10) */
class MyClass {}
```

コンパイラ エラー VJS1403

エラー メッセージ
名前付きフィールドまたはパラメータ '<パラメータ名>' は、割り当てられません。

名前付きパラメータは、読み取り専用であるため初期化できません。

使用例

```
// VJS1403a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttribute : System.Attribute
{
   public MyAttribute () {}
   public readonly int Version = 20;
}
```

```
// VJS1403b.jsl
// compile with: /target:library /reference:VJS1403a.dll
/** @attribute MyAttribute (Version = 10) */   // VJS1403
// try the following line instead
// /** @attribute MyAttribute () */
class MyClass {}
```

コンパイラ エラー VJS1404

エラー メッセージ
型 '<型 1>' の値を、型 '<型 2>' の属性フィールドまたはプロパティに割り当てられません。

フィールドに代入する値の型が正しくありません。

使用例

```
// VJS1404a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttribute : System.Attribute
{
   public MyAttribute () {}
   public int Version = 20;
}
```

```
// VJS1404b.jsl
// compile with: /target:library /reference:VJS1404a.dll
/** @attribute MyAttribute (Version = "string") */   // VJS1404
// try the following line instead
// /** @attribute MyAttribute (Version = 10) */
class MyClass {}
```

コンパイラ エラー VJS1405

エラー メッセージ
名前付きフィールドまたはプロパティ '<パラメータ名>' は、静的です。

スタティックな名前付きフィールド パラメータは使用できません。

使用例

```
// VJS1405a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttribute : System.Attribute
{
   public MyAttribute() {}
   public static int Version = 20;
   public int Version2 = 20;
}
```

次の例では、VJS1405 エラーが生成されます。

```
// VJS1405b.jsl
// compile with: /target:library /reference:VJS1405a.dll
/** @attribute MyAttribute (Version = 10) */   // VJS1405
// try the following line instead
// /** @attribute MyAttribute (Version2 = 10) */
class MyClass {}
```

コンパイラ エラー VJS1406

エラー メッセージ
名前付きフィールドまたはプロパティ '<パラメータ名>' は、パブリックではありません。

プライベートな名前付きパラメータにはアクセスできません。

次の例では、VJS1406 エラーが生成されます。

使用例

```
// VJS1406a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttribute : System.Attribute
{
   private int Version = 1;
   public int Version2 = 1;

   public MyAttribute()
   {
      int i = Version;
   }
}
```

```
// VJS1406b.jsl
// compile with: /target:library /reference:VJS1406a.dll
/** @attribute MyAttribute (Version = 10) */   // VJS1406
class MyClass {}

/** @attribute MyAttribute (Version2 = 10) */   // OK
class MyClass2 {}
```

コンパイラ エラー VJS1407

エラー メッセージ
System.AttributeUsageAttribute は System.Attribute から引き継がれたクラスにのみアタッチします。

System.AttributeUsageAttribute は、すべてのシステム属性またはユーザー定義属性に追加される特殊な属性です。この属性は、**System.Attribute** から拡張される型に適用されます。

Visual J# では属性を作成できないため、この属性はどの型にも適用できません。

次の例では、VJS1407 エラーが生成されます。

```
// VJS1407.jsl
// compile with: /target:library
// VJS1407 expected
/** @attribute System.AttributeUsage(System.AttributeTargets.All) */
class MyClass
{
}
```

コンパイラ エラー VJS1408

エラー メッセージ
属性 'type' は、System.AttributeUsageAttribute を使用して明示的に設定されなければなりません

このエラーは、System.AttributeUsageAttribute で宣言されていない属性を使用した場合に発生します。これは特殊な属性であり、Visual J# のすべての属性クラスで必要になります。

次の例では、VJS1408 エラーが生成されます。

```
// VJS1408.jsl

// Remove the first '//' comment symbols on the next line to avoid this error:
// /** @attribute System.AttributeUsage(System.AttributeTargets.All) */
public class MyAttribute extends System.Attribute
{
   public MyAttribute()
   { }
}


/** @attribute MyAttribute() */  // VJS1408
public class MyAttributeClass
{
   public static void main(String [] args)
   {
   }
}
```

コンパイラ エラー VJS1409

エラー メッセージ
属性 'type' は同じ宣言の中で何度も使用できません。

属性が 2 回以上指定されています。

使用例

```
// VJS1409a.cs
// compile with: /target:library
// a C# program
[System.AttributeUsage(System.AttributeTargets.All)]
public class MyAttr : System.Attribute {}

[System.AttributeUsage(System.AttributeTargets.All, AllowMultiple = true )]
public class MyAttr2 : System.Attribute {}
```

次の例では、VJS1409 エラーが生成されます。

```
// VJS1409b.jsl
// compile with: /target:library /reference:VJS1409a.dll
/** @attribute MyAttr() */
/** @attribute MyAttr() */   // VJS1409
class MyClass {}

// OK
/** @attribute MyAttr2() */
/** @attribute MyAttr2() */
class MyClass2 {}
```

コンパイラ エラー VJS1410

エラー メッセージ
属性 'type' はこの宣言で有効ではありません。

属性が、意図していない構成要素に適用されました。

次の例では、VJS1410 エラーが生成されます。

```
//VJS1410.jsl
// compile with: /target:library
/** @assembly System.Obsolete ("obsolete") */   // VJS1410
class MyClass {}
```

コンパイラ エラー VJS1413

エラー メッセージ
属性 'type1' は、パブリックでない 'type2' を拡張します

このエラーは、属性を定義するクラスで、パブリックとして宣言されていないクラスを拡張した場合に発生します。属性を定義する場合、継承の階層構造に含まれるすべてのクラスはパブリックとして宣言されている必要があります。

次の例では、VJS1413 エラーが生成されます。

```
// VJS1413.jsl
import System.*;

class BaseAttribute extends System.Attribute
{
  public BaseAttribute() {}
}

/**@attribute AttributeUsageAttribute(AttributeTargets.All,Inherited = true)*/
public class DerivedAttribute extends BaseAttribute  // VJS1413
{
   public DerivedAttribute (String s)
   {}
}

class CMain
{
   public static void main(String [] args) {}
}
```

コンパイラ エラー VJS1418

エラー メッセージ
@conditional ディレクティブは、戻り型が void のメソッドでのみ有効です。

@conditional ディレクティブは、非 void メソッドでは使用できません。

次の例では、VJS1418 エラーが生成されます。

```
// VJS1418.jsl
// compile with: /target:library
class MyClass
{
  /** @conditional(DEBUG) */   // VJS1418
   public int Test1()
   {
      return 0;
   }
   // try the following instead
   /** @conditional(DEBUG)*/
   public void Test2()
   {
   }
}
```

コンパイラ エラー VJS1420

エラー メッセージ
デリゲートの作成は、引数を 1 つまたは 2 つ持たなければなりません。

デリゲート コンストラクタで使用できるパラメータは、1 つまたは 2 つに限られます。デリゲートの有効なコンストラクタを次に示します。

- *MyDelegate* **(** *method* **)**
- *MyDelegate* **(** *Object,String* **)**

次の例では、VJS1420 エラーが生成されます。

```
// VJS1420.jsl
// compile with: /target:library
class MyClass
{
   delegate void MyDel();
   public void Test()
   {
      MyDel d = new MyDel(new MyClass (), "DelMethod", null);   // VJS1420

      // OK. param is a method
      MyDel d1 = new MyDel(DelMethod);

      // OK. passing object and the method name
      MyDel d2 = new MyDel(new MyClass (), "DelMethod");
   }
   public void DelMethod()
   {
   }
}
```

コンパイラ エラー VJS1421

エラー メッセージ
デリゲートの作成に本体を含むことはできません。

Visual J# コンパイラでは、デリゲートの作成がサポートされていません。

次の例では、VJS1421 エラーが生成されます。

```
// VJS1421.jsl
class MyDelegate extends System.Delegate   // VJS1421
{
}
```

コンパイラ エラー VJS1422

エラー メッセージ
デリゲートの作成 '<デリゲート名>' で、2 番目の引数は文字列でなければなりません。

デリゲートを作成する場合は、2 番目の引数を文字列にする必要があります。

次の例では、VJS1422 エラーが生成されます。

```
// delegate creation second arg must be a string
// VJS1422.jsl
// compile with: /target:library
class MyClass
{
   delegate void MyDel();
   public void Test()
   {
      MyDel d = new MyDel(new MyClass (), new Object());   // VJS1422

      // OK, passing object and the method string
      MyDel d2 = new MyDel (new MyClass (), "DelMethod");
   }

   public void DelMethod()
   {
   }
}
```

コンパイラ エラー VJS1423

エラー メッセージ
デリゲート '<デリゲート名>' の作成で、引数はメソッドを指定しなければなりません。

デリゲートを作成するには、メソッドが必要です。

次の例では、VJS1423 エラーが生成されます。

```
// VJS1423.jsl
// compile with: /target:library
class MyClass
{
   delegate void MyDelegate();
   public void Test()
   {
      MyDelegate d = new MyDelegate (new MyClass ());   // VJS1423
      MyDelegate d1 = new MyDelegate (new MyClass ().DelMethod);   // OK
   }
   public void DelMethod()
   {
   }
}
```

コンパイラ エラー VJS1424

エラー メッセージ
'<メソッド名>' の戻り値の型は '<型>' でなければなりません。

デリゲートの戻り値の型が、デリゲートに関連付けられたメソッドの戻り値の型と一致しません。

次の例では、VJS1424 エラーが生成されます。

```
// VJS1424.jsl
// compile with: /target:library
class MyClass
{
   delegate void MyDelegate();
   public void Test1()
   {
      MyDelegate d = new MyDelegate (new MyClass ().Test2);   // VJS1424

      // try the following line instead
      MyDelegate d1 = new MyDelegate (new MyClass ().Test3);
   }
   public int Test2()
   {
      return 0;
   }
   public void Test3()
   {
   }
}
```

コンパイラ エラー VJS1425

エラー メッセージ
メソッド '<メソッド名>' のシグネチャは '<デリゲート名>' デリゲートのシグネチャと一致しません。

デリゲートのシグネチャが、デリゲートに関連付けられたメソッドのシグネチャと一致しません。

次の例では、VJS1425 エラーが生成されます。

```
// VJS1425.jsl
// compile with: /target:library
class MyClass
{
   delegate void MyDelegate(int i);
   public void Test1()
   {
      MyDelegate d = new MyDelegate(new MyClass ().Test2);   // VJS1425
      // try the following line instead
      MyDelegate d1 = new MyDelegate(new MyClass ().Test3);
   }

   void Test2(long l)
   {
   }
   void Test3(int i)
   {
   }
}
```

コンパイラ エラー VJS1426

エラー メッセージ
静的メソッド '<メソッド名>' は、デリゲートの初期化には使用できません。

デリゲートを作成する場合にスタティック メソッドを渡すことはできません。使用できるのはインスタンス メソッドだけです。

次の例では、VJS1426 エラーが生成されます。

```
// VJS1426.jsl
// compile with: /target:library
class MyClass
{
   delegate void MyDelegate();
   public void Test1()
   {
      MyDelegate d = new MyDelegate(Test2);   // VJS1426
      // try the following line instead
      MyDelegate d1 = new MyDelegate(Test3);
   }

   static void Test2()
   {
   }
   void Test3()
   {
   }
}
```

コンパイラ エラー VJS1427

エラー メッセージ
.NET デリゲート '<デリゲート名>' 作成の引数はリテラル文字列でなければなりません。

.NET Framework のデリゲートを作成する場合は、2 番目の引数をリテラル文字列にする必要があります。

使用例

```
// VJS1427a.cs
// compile with: /target:library
// a C# program
public delegate void MyDelegate();
```

次の例では、VJS1427 エラーが生成されます。

```
// VJS1427b.jsl
// compile with: /target:library /reference:VJS1427a.dll
class MyClass
{
   public void Test1()
   {
      String s = "m1";
      MyDelegate d = new MyDelegate(new MyClass (), s);   // VJS1427
      MyDelegate d1 = new MyDelegate(new MyClass (), "Test2");   // OK
   }

   void Test2() {}
}
```

コンパイラ エラー VJS1428

エラー メッセージ
メソッド '<メソッド名>' は、デリゲートの throws 句にない '<例外名>' をスローします。

デリゲート宣言で例外が指定されていない場合、デリゲートに割り当てられたメソッドは例外をスローできません。

次の例では、VJS1428 エラーが生成されます。

```
// or declare the delegate to throw
// VJS1428.jsl
// compile with: /target:library
class MyClass
{
   delegate void MyDelegate();
   public void Test1()
   {
      MyDelegate d = new MyDelegate(new MyClass ().Test2);   // VJS1428
      // try the following line instead
      MyDelegate d1 = new MyDelegate(Test3);
   }

   void Test2() throws Exception
   {
   }
   void Test3()
   {
   }
}
```

コンパイラ エラー VJS1432

エラー メッセージ
'type' は、final として明示的に宣言されなければなりません

Visual J# では、値型を派生させることはできないため、値型のクラスを定義する場合、**final** キーワードが必要です。詳細については、「ユーザー定義の値型」を参照してください。

次の例では、VJS1432 エラーが生成されます。

```
// VJS1432.jsl
import System.*;

// Try this instead:
// public final class MyValueType extends System.ValueType
public class MyValueType extends System.ValueType // VJS1432
{
}

public class CMain
{
    public static void main() {}
}
```

コンパイラの警告 (レベル 1) VJS1434

エラー メッセージ
'@<属性>' を無視します -- 拡張機能は無効です。

@dll.import、**@attribute**、**@com**、**delegate** などのキーワードは、標準 Java 言語の拡張機能です。/x:all を指定してコンパイルした場合は、すべての拡張機能が無効になります。拡張機能は無視されます。

次の例では、VJS1434 エラーが生成されます。

```
// VJS1434.jsl
// compile with: /target:library /x:all /W:1
/** @dll.import("user32") */   // VJS1434
class MyClass
{
}
```

コンパイラ エラー VJS1435

エラー メッセージ
'<クラス名>' 内部クラスの内部で、デリゲートを宣言することはできません。

デリゲートは、内部クラスのメンバとして宣言できません。

次の例では、VJS1435 エラーが生成されます。

```
// VJS1435.jsl
// compile with: /target:library

class MyClass
{
   class InnerClass
   {
      delegate void del();   // VJS1435
   }

   delegate void MyDelegate1();   // OK, declare as an outer class member
}

delegate void MyDelegate2 ();   // OK, declare delegate as separate member
```

コンパイラ エラー VJS1436

エラー メッセージ
null リテラルへのメソッドの呼び出しはできません。

null 式でのメソッド呼び出しは、実行時に必ず失敗します。

次の例では、VJS1436 エラーが生成されます。

```
// VJS1436.jsl
class MyClass
{
   public static void main (String [] args)
   {
      null.toString();   // VJS1436, method call on null literal
   }
}
```

コンパイラ エラー VJS1437

エラー メッセージ
メソッド呼び出しをここで実行することはできません。

定数文字列は、パラメータとしてだけ渡すことができます。

次の例では、VJS1437 エラーが生成されます。

```
// VJS1437.jsl
// compile with: /target:library
/** @assembly System.Reflection.AssemblyTitle("abc".ToString()) */   // VJS1437
// try the following line instead
// /** @assembly System.Reflection.AssemblyTitle("abc") */
class MyClass
{
}
```

コンパイラ エラー VJS1438

エラー メッセージ
静的なコンテキストで、静的でないメンバ '<名前>' にアクセスすることはできません。

静的コンストラクタや静的メンバ関数では、クラスのインスタンス メンバにアクセスできません。

次の例では、VJS1438 エラーが生成されます。

```
// VJS1438.jsl
// compile with: /target:library
// VJS1438
class MyClass
{
   int name = 0;
   public static int function()
   {
      int i = 0;
      i = name; //error attemting to reference instance member "name"
      return i;
   }
}
```

このエラーを解決するには、**name** 変数を静的変数にするか、**function** を非静的にする必要があります。

コンパイラ エラー VJS1439

エラー メッセージ
配列初期化子が正しく作成されていません。

混合配列の初期化が無効です。配列の各要素は、個別に初期化する必要があります。混合配列の配列の初期化が無効です。配列の各要素は、個別に初期化する必要があります。

次の例では、VJS1439 エラーが生成されます。

```
// VJS1439.jsl
// compile with: /target:library
class Myclass
{
   void f()
   {
      int [,][] arr = new int [10,10][10];   // VJS1439
   }
}
```

代わりに、次の正しいコードを使ってください。

```
// VJS1439a.jsl
// compile with: /target:library
class Myclass
{
   void f()
   {
      // Remove the last dimension and explicitly initialize
      // the rectangular array.
      int [,][] arr = new int [10,10][];
      for (int i = 0; i < 100; i++)
         for (int j = 0; j < 100; j++)
            arr[i,j] = new int[10];
   }
}
```

コンパイラ エラー VJS1440

エラー メッセージ
メンバ': '型' はプロテクト メンバを含むことができません

値型のメンバ宣言で **protected** 修飾子を使用することはできません。値型は派生させることができないので、**protected** を指定する意味がないためです。

次の例では、VJS1440 エラーが生成されます。

```
// VJS1440.jsl
// compile with: /target:library
import System.*;

public final class MyValueType extends System.ValueType
{

   public int i;
   private int j;
   protected int k;  // VJS1440

   public MyValueType(int _i, int _j, int _k)
   {
     i = _i;
     j = _j;
     k = _k;
   }

}
```

コンパイラ エラー VJS1441

エラー メッセージ
'型' に明示的なパラメータのないコンストラクタを含めることはできません。

このエラーは、値型のクラス (**System.ValueType** を継承するクラス) が、引数を持たないコンストラクタで定義されている場合に発生します。値型は構築時に完全に定義されることが必要です。引数を持たないコンストラクタを使用することはできません。Visual J# における値型の定義の詳細については、「ユーザー定義の値型」を参照してください。

次の例では、VJS1441 エラーが生成されます。

```
// VJS1441.jsl
import System.*;

public final class MyValueType extends System.ValueType
{
   int m_i;

   public MyValueType()  // VJS1441
   {
   }
   // Try this instead:
   // public MyValueType(int i)
   // {
   //     m_i = i;
   // };
}

class CMain
{
   public static void main() {}
}
```

コンパイラ エラー VJS1442

エラー メッセージ
値型のクラスのコンストラクタで 'super' の明示的な呼び出しを指定できません

System.ValueType を拡張するクラスのコンストラクタでは、super を明示的に呼び出すことはできません。値型ではスーパー クラスのコンストラクタ呼び出しは無効です。

次の例では、VJS1442 エラーが生成されます。

```
// VJS1442.jsl
// compile with: /target:library
import System.*;

public final class MyValueType extends System.ValueType
{

   public MyValueType(int z)
   {
      super();  // VJS1442
   }
}
```

コンパイラ エラー VJS1443

エラー メッセージ
Value Type 型の内部で instance 初期化子のブロックを指定できません

値型でインスタンス初期化子ブロックを使用することはできません。インスタンス初期化子ブロックとは、関数本体に属していない、クラス内のコード ブロックをいいます。インスタンス初期化子ブロックは、クラスのインスタンスが作成されるときに、コンストラクタの前に実行されます。

次の例では、VJS1443 エラーが生成されます。

```
// VJS1443.jsl
import System.*;

public final class MyValueType extends System.ValueType
{
   public int integer;
   public int[] ia;
   // instance initializer:
   {                                                    // VJS1443
      ia = new int[100];
      for (int k = 0; k < 100; k++)
         ia[k] = k;
   }

   public MyValueType(int i)
   {
      integer = i;
   }
}

class CMain
{
   public static void main()
   {
      MyValueType mvt = new MyValueType();
   }
}
```

コンパイラ エラー VJS1444

エラー メッセージ
列挙型 '型' でインターフェイスを実装できません

列挙型 (Enum) は通常のクラスとは異なるため、インターフェイスを実装することはできません。

このエラーを解決するには

- implements キーワードおよびすべてのインターフェイス名を削除します。

使用例
次の例では、VJS1444 エラーが生成されます。

```
// VJS1444.jsl
// Compile with: /target:libary
// VJS1444 expected
import System.*;

enum MyEnum implements MyInterface
{

}

public interface MyInterface
{

}
```

参照
関連項目
ユーザー定義の列挙型 (J#)
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1445

エラー メッセージ
列挙型でメソッドを宣言できません

列挙型 (Enum) はメソッドを持つことができないなど、通常のクラスとは異なります。

このエラーを解決するには

- Enum クラスで定義されたすべてのメソッドを削除します。

使用例

次の例では、VJS1445 エラーが生成されます。

```
// VJS1445.jsl
import System.*;

enum MyEnum
{
   Red(10);
   public void method ( )  // VJS1445
   {
   }

}
```

参照

関連項目

ユーザー定義の列挙型 (J#)

その他の技術情報

Visual J# コンパイラ

コンパイラ エラー VJS1446

エラー メッセージ
'型' にファイナライザを含めることはできません

Enum は値型であり、宣言元のスタック フレームが破棄されると、必ず破棄されます。Finalize は、ヒープ上のオブジェクトに対してのみ有効であるため、Enum で使用することはできません。

次の例では、VJS1446 エラーが生成されます。

```
// VJS1446
import System.*;

public final class MyValueType extends System.ValueType
{
     public void Finalize() { }  // VJS1446
}

class CMain
{
     public static void main() { }
}
```

コンパイラ エラー VJS1448

エラー メッセージ
ValueType には '修飾子' 修飾子を指定できません。

abstract キーワードを削除します。値型を抽象型にすることはできません。

次の例では、VJS1448 エラーが生成されます。

```
// VJS1448.jsl
import System.*;

public abstract class MyValueType extends System.ValueType  // VJS1448
{

}
```

コンパイラ エラー VJS1449

エラー メッセージ
'フィールド宣言' : '型' にインスタンス フィールド初期化子を指定することはできません。

値型では、インスタンス フィールドを特定の値で初期化することはできません。値型の既定値は、すべてのメンバを既定値に設定することによって決定されます。それ以外の既定値は使用できません。

次の例では、VJS1449 エラーが生成されます。

```
// VJS1449.jsl
import System.*;


public final class MyValueType extends System.ValueType
{
    public long ONE   = 1;  // VJS1449
}

public class CMain
{
    public static void main()
    {
    }
}
```

コンパイラの警告 (レベル 2) VJS1450

エラー メッセージ
識別子 'identifier' は CLS に準拠していません。

CLS 準拠のクラスでは、アンダースコア文字で始まる識別子を使用することはできません。

次の例では、VJS1450 エラーが生成されます。

```
// VJS1450.jsl
// compile with: /target:library /W:2
/** @assembly System.CLSCompliant(true) */

// Because the assembly level CLSCompliant attribute
// is true, this type will be checked for compliance
public class MyClass
{
   public int _value;    // VJS1450
   public int value;   // OK
}
```

参照

関連項目
共通言語仕様への準拠の検証
概念
共通言語仕様

コンパイラの警告 (レベル 2) VJS1451

エラー メッセージ
大文字、小文字の違いのみの識別子 'identifier_1' および 'identifier_1' は、CLS に準拠していません。

CLS 準拠が有効にされており、1 つのスコープ内に大文字小文字だけが異なる識別子が複数存在する場合に、このエラーが発生します。これは Visual J# では許可されますが、CLS には準拠していません。エラーを解決するには、変数名を変更するか、関連する型の CLS 準拠を無効にします。

次の例では、VJS1457 エラーが生成されます。

```
// VJS1451.jsl
// compile with: /target:library /W:2
/** @assembly System.CLSCompliant(true) */
public class MyClass
{
   public int value;   // VJS1451
   public int VALUE;
}
```

参照

関連項目
共通言語仕様への準拠の検証
概念
共通言語仕様

コンパイラの警告 (レベル 2) VJS1452

エラー メッセージ
クラス 'クラス' には識別子 '識別子' の定義が既に含まれています

同じ名前がオーバーロードによって解決される場合を除き、CLS 準拠のスコープで使用した名前はすべて、その種類に関係なく一意に識別できることが必要です。

使用例

アセンブリ レベル CLSCompliant 属性は true なので、この型は準拠しているかどうかを確認されます。次の例では、VJS1452 エラーが生成されます。

```
// VJS1452.jsl
// compile with: /target:library /W:2
/** @assembly System.CLSCompliant(true) */

public class MyClass
{
   public int val;   // VJS1452
   public void val(int i) {}
}
```

参照

関連項目

共通言語仕様への準拠の検証

概念

共通言語仕様

コンパイラ エラー CLS VJS1457

エラー メッセージ
Super クラスまたはインターフェイス '<型>' は CLS 準拠ではありません。

Java 言語の型の派生元である型が、共通言語仕様 (CLS: Common Language Specification) に準拠していません。

使用例

```
// VJS1457a.cs
// compile with: /unsafe /target:library
public abstract class Unsafe
{
   public abstract unsafe void UnsafeMethod(byte * src);
}
```

```
// vjs1457b.jsl
// compile with: /reference:vjs1475a.dll
public class MyClass extends Unsafe {}   // VJS1457
```

参照

関連項目
共通言語仕様への準拠の検証
概念
共通言語仕様

コンパイラ エラー CLS VJS1459

エラー メッセージ
'フィールド' フィールドの型は CLS 準拠ではありません。

CLS 準拠の型では、パブリック フィールドの型を CLS 準拠にする必要があります。.NET Framework には、**System.Byte** など、代用可能な CLS 準拠の型が用意されています。CLS 準拠の型として代用可能な型の一覧については、「共通言語仕様」を参照してください。

次の例では、VJS1459 エラーが生成されます。

```
// VJS1459.jsl
// compile with: /target:library /W:2
/** @assembly System.CLSCompliant(true) */
public class MyClass
{
   public byte value;   // VJS1459
}
```

参照

関連項目

共通言語仕様への準拠の検証

コンパイラの警告 (レベル 2) VJS1460

エラー メッセージ
メソッド 'メソッド' の戻り値の型は CLS に準拠していません。

CLS 準拠の型では、メソッドの戻り値の型を CLS 準拠にする必要があります。.NET Framework には、**System.Byte** など、代用可能な CLS 準拠の型が用意されています。CLS 準拠の型として代用可能な型の一覧については、「共通言語仕様」を参照してください。

次の例では、VJS1460 エラーが生成されます。

```
// VJS1460.jsl
// compile with: /W:2 /target:library
/** @assembly System.CLSCompliant(true) */
public class MyClass
{
   public byte method() { return (byte) 1; }  // VJS1460
}
```

参照

関連項目

共通言語仕様への準拠の検証

コンパイラの警告 (レベル 2) VJS1461

エラー メッセージ
メソッド 'メソッド' のパラメータ型は CLS に準拠していません

CLS 準拠の型では使用できない型がパラメータに使用されています。.NET Framework には、**System.Byte** など、代用可能な CLS 準拠の型が用意されています。CLS 準拠の型として代用可能な型の一覧については、「共通言語仕様」を参照してください。

次の例では、VJS1461 エラーが生成されます。

```
// VJS1461.jsl
// compile with: /W:2
/** @assembly System.CLSCompliant(true) */

// Because the assembly level CLSCompliant attribute
// is true, this type will be checked for compliance
public class MyClass
{
   public void method(byte sb) {}    // VJS1461
   public static void main(String [] args) {}
}
```

参照

関連項目
共通言語仕様への準拠の検証
概念
共通言語仕様

コンパイラの警告 (レベル 2) VJS1463

エラー メッセージ
静的メソッドを定義しているインターフェイス 'インターフェイス' は CLS に準拠していません

インターフェイスでは、CLS 準拠の型としてフィールドを指定することはできません。

使用例

アセンブリ レベル CLSCompliant 属性は true なので、この型は準拠しているかどうかを確認されます。次の例では VJS1463 エラーが生成されます。

```
// VJS1463.jsl
// compile with: /target:library /W:2
/** @assembly System.CLSCompliant(true) */
interface IA
{
   int i = 0;   // VJS1463
}
```

参照

関連項目
共通言語仕様への準拠の検証
概念
共通言語仕様

コンパイラ エラー CLS VJS1465

エラー メッセージ
型は、シグネチャよりもアクセス可能です

このエラーは、指定された型のアクセス レベルが、親よりも低いことを示しています。このエラーは、**System.CLSCompliant** を **true** に設定し、CLS 準拠としてコードを指定した場合に発生することがあります。

このエラーを解決するには

- *type* のシグネチャが、親クラスに基づく適切なアクセス レベルを持っていることを確認します。

使用例
次の例では、CLS VJS1465 エラーが生成されます。

```
import System.*;
/** @assembly System.CLSCompliant(true) */
class B
{
/** @attribute System.CLSCompliant(true) */
public class BInner
{

}
}
```

参照
関連項目
共通言語仕様への準拠の検証
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1467

エラー メッセージ
System.TypedReference は、ローカル変数およびパラメータに対してのみ有効な型です。

TypedReference が、無効な場所で使用されました。

次の例では、VJS1467 エラーが生成されます。

```
// VJS1467.jsl
class MyClass
{
   System.TypedReference typeRef;   // VJS1467
   public static void main (String [] args)
   {
      System.TypedReference localRef;   // OK on local var
   }
   public void m (System.TypedReference paramRef)   // OK on param
   {
   }
}
```

コンパイラ エラー VJS1468

エラー メッセージ
アセンブリの属性で指定されたアセンブリ、またはファイルのバージョン '<バージョン番号>' は無効です。

アセンブリのバージョン番号の形式は major.minor.build.revision で、各番号はすべて 0 を超える値であることが必要です。

次の例では、VJS1468 エラーが生成されます。

```
// VJS1468.jsl
// compile with: /target:library
import System.Reflection.*;
/** @assembly AssemblyVersion ("1.-1.1.1") */   // VJS1468
class MyClass
{
}
```

コンパイラ エラー VJS1469

エラー メッセージ
アセンブリは遅延署名されていません。

VJS1469 エラーは、指定したキー ファイルまたはキー コンテナに問題がある場合に、発生する可能性があります。アセンブリに完全署名するには、公開キーと秘密キーに関する情報を含む有効なキー ファイルを提供する必要があります。アセンブリに遅延署名するには、[遅延署名のみ] チェック ボックスをオンにし、公開キーの情報を含む有効なキー ファイルを提供する必要があります。アセンブリに遅延署名する場合、秘密キーは必要ありません。詳細については、「方法 : アセンブリに署名する (Visual Studio)」を参照してください。

sn -p を使う場合は、公開キーだけを含むキーファイルを作成できます。

遅延署名については、「/delaysign (暗号化キーの遅延署名)」を参照してください。

次の例では、VJS1469 エラーが生成されます。

```
// VJS1469.jsl
import System.Reflection.*;
import System.Runtime.CompilerServices.*;
/** @assembly AssemblyDelaySign(false) */
// Assume VJS1469.key only has public key
/** @assembly AssemblyKeyFile("VJS1469.key") */ // VJS1469

class MyClass
{
   public static void main (String [] args) {}
}
```

コンパイラ エラー VJS1470

エラー メッセージ
無効なキー ファイル '<ファイル名>'

AssemblyKeyFileAttribute 属性に渡されたファイル名が、ディスクで見つかりませんでした。または、このファイルに有効なキー ペアや公開キーが含まれていませんでした。[署名] ページの詳細については、「[署名] ページ (プロジェクト デザイナ)」を参照してください。

次の例では、VJS1470 エラーが生成されます。

```
// VJS1470.jsl
import System.Reflection.*;
import System.Runtime.CompilerServices.*;

/** @assembly AssemblyKeyFile("file") */   // VJS1470
class MyClass
{
   public static void main (String [] args)
   {
   }
}
```

参照

処理手順
方法 : 公開キーと秘密キーのキー ペアを作成する
関連項目
[署名] ページ (プロジェクト デザイナ)

コンパイラ エラー VJS1471

エラー メッセージ
無効なキー名 '<コンテナ名>' です。

AssemblyKeyNameAttribute 属性に渡されたキー名は無効です。

次の例では、VJS1471 エラーが生成されます。

```
// VJS1471.jsl
import System.Reflection.*;
import System.Runtime.CompilerServices.*;

/** @assembly AssemblyKeyName("container") */   // VJS1471
class MyClass
{
   public static void main (String [] args)
   {
   }
}
```

コンパイラ エラー VJS1472

エラー メッセージ
'<型 1>' は有効なフィールドまたはメソッド宣言ではありません。'<型 2>' を使用してください。

'<型 1>' は、フィールドまたはメソッド宣言として使用できません。このエラーを解決するには、'<型 2>' を使います。

次の例では、VJS1472 エラーが生成されます。

```
// VJS1472.jsl
// compile with /target:library
// VJS1472 expected
public class MyClass
{
   void f(System.Int32 int32) {}
}
```

コンパイラ エラー VJS1473

エラー メッセージ
'member' : CLS 準拠のインターフェイスは CLS 準拠メンバを持たなければなりません。

インターフェイス メンバを非 CLS 準拠にすることはできません。CLS では、すべてのインターフェイス メンバが実装されるように指定します。

次の例では、VJS1473 エラーが生成されます。

```
// VJS1473.jsl
// compile with: /target:library /W:2
/** @assembly System.CLSCompliant(true) */

public interface IA
{
   /** @attribute.method System.CLSCompliant(false) */
   public void m1();   // VJS1473
}
```

コンパイラの警告 (レベル 2) VJS1474

エラー メッセージ
'member' : CLS 準拠メンバだけが抽象になることができます。

クラス メンバに abstract と CLS 非準拠の両方を指定することはできません。CLS では、すべてのクラス メンバが実装されるように指定します。

次の例では、VJS1474 エラーが生成されます。

```
// VJS1474.jsl
// compile with: /target:library /W:2
/** @assembly System.CLSCompliant(true) */
public abstract class MyClass
{
   /** @attribute.method System.CLSCompliant(false) */
   public abstract void method2();    // VJS1474
}
```

コンパイラの警告 (レベル 2) VJS1475

エラー メッセージ
private であるため、CLS 準拠の確認は 'メンバ' で実行されます

CLS の規則は、それらが定義されているアセンブリの外部から参照可能な型またはメンバにのみ適用されます。

次の例では、VJS1475 エラーが生成されます。

```
// VJS1475.jsl
// compile with: /target:library /W:2
/** @assembly System.CLSCompliant(true) */
 public class MyClass
{

    /** @attribute System.CLSCompliant(true) */
    private int i;        // VJS1475
}
```

コンパイラ エラー CLS VJS1477

エラー メッセージ
プロパティ 'プロパティ' は、CLS に準拠していません。

このエラーは、CLS 準拠のチェック対象となるソースに、bean のプロパティが使用されている場合に発生します。このエラーを解決するには、CLS 準拠のコードでは bean のプロパティを使わないようにします。

次の例では、CLS VJS1477 エラーが生成されます。

使用例
次の例では、VJS1477 エラーが生成されます。

```
/**@assembly System.CLSCompliantAttribute(true) */

public class AssemblyCompliant
{
protected int fprop = 23;

/**@beanproperty*/ public void setprop(int val)
{
fprop = val;
}

/**@beanproperty*/ public int getprop()
{
return fprop ;
}
}
```

参照
概念
CLS 準拠コードの記述
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1478

エラー メッセージ
'型' は CLS 準拠型のみを使用するコンストラクタにアクセスできません。

このエラーは、カスタム属性に配列パラメータが存在する場合に発生します。

このエラーを解決するには

- このエラーを解消するには、コンストラクタで他の型のパラメータを使用します。

使用例
次の例では VJS1478 エラーが生成されます。

```
// VJS1478.jsl
// compile with: /target:library
// VJS1478 expected
/**@assembly System.CLSCompliantAttribute(true)*/
class MyAttrib extends System.Attribute
{
   public MyAttrib(int[] arg) {}   // VJS1478
   // try the following line instead
   // public MyAttrib() {}
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラの警告 (レベル 3) VJS1482

エラー メッセージ
最終フィールド 'フィールド' には、static を宣言できます。

この警告は、オブジェクトの各インスタンスについて領域を確保するのではなく、final フィールドを static として宣言することで効率を高めることができることを示しています。

このエラーを解決するには

- この警告を回避するには、フィールドの宣言に static キーワードを追加します。

使用例

次のコードのコメントを解除すると、VJS1482 エラーが生成されます。`//public final int i = 10;`

```
// vjs1482.jsl
// compile with: /W:3
public class C
{
public final int i = 10;
//public static final int i = 10;  // more efficient

public static void main()
   {
   }
}
```

参照

その他の技術情報

Visual J# コンパイラ

コンパイラの警告 (レベル 3) VJS1483

エラー メッセージ
空の try ステートメントです

この警告は、コード内に空の **try** ステートメントが存在することを示しています。

このエラーを解決するには

- この警告を回避するには、**try** ステートメントに必ずなんらかの処理を記述します。

使用例

次の例では、VJS1483 エラーが生成されます。

```
public class Test
{
   public static void main()
   {
      try { }
      catch (Exception ex) { }
      try { System.out.println("Hi"); }
      finally { }
   }
```

参照

その他の技術情報

Visual J# コンパイラ

コンパイラの警告 (レベル 3) VJS1484

エラー メッセージ
空の finally ステートメントです

この警告は、コード内に空の **finally** ステートメントが存在することを示しています。

このエラーを解決するには

- この警告を回避するには、**finally** ステートメントに必ずなんらかの処理を記述します。

使用例
次の例では、VJS1484 エラーが生成されます。

```
public class Test {
    public static void main() {

       try { } catch (Exception ex){}


       try { System.out.println("Hi");} finally {}
    }
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラの警告 (レベル 3) VJS1485

エラー メッセージ
System.Exception の catch 句がすべての例外をキャッチします。Throwable の別の句は必要ありません

Throwable 句および **System.Exception catch** 句がコード内で明示的に指定されています。

このエラーを解決するには

- Java 言語のように、すべての例外に該当する **Throwable** をキャッチするのではなく、**Throwable** から派生した **System.Exception** をキャッチします。

使用例
次の例では、VJS1485 エラーが生成されます。

```
public class CatchSysExAfterCatchTh
{
   public static void main(String[] args)
   {
      try
      {
         throw new System.SystemException();
      }
      catch (Throwable t)
      {
      }
      catch (System.Exception nfe)
      {
      }
   }
}
```

参照
関連項目
Visual J# の例外の階層構造
その他の技術情報
Visual J# コンパイラ

コンパイラの警告 (レベル 3) VJS1486

エラー メッセージ
Throwable の catch 句は、この句の後に記述しなければなりません。

catch 句が指定されていますが、その後ろに必要な Throwable が存在しません。

このエラーを解決するには

1. コード内で使用されている **catch** ステートメントをチェックし、後続の **Throwable** が欠落している **catch** 句を探します。
2. 適切な **catch** 句を追加します。

使用例
次の例では、VJS1486 エラーが生成されます。

```
public class CatchUnmapdExAfterCatchTh
{
public static void main(String[] args)
{
try
{
throw new System.SystemException();
}
catch (Throwable t)
{
}
catch (System.SystemException nfe)
{
}
}
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラの警告 (レベル 3) VJS1493

エラー メッセージ
変数 '<変数名>' が使用されていません。

宣言された変数が、初期化および使用されていません。

次の例では、VJS1493 エラーが生成されます。

```
// VJS1493.jsl
// compile with: /W:3
public class MyClass
{
   public static void main()
   {
      int i;   // VJS1493, i is not initialized or used
   }
}
```

コンパイラの警告 (レベル 1) VJS1499

エラー メッセージ
'private protected' はサポートされていません。'protected' を使用します。

アクセス指定子 **private protected** はサポートされていません。代わりに、**protected** を使ってください。

次の例では、VJS1499 エラーが生成されます。

```
// VJS1499.jsl
// compile with: /W:1
public class MyClass
{
   private protected void Test()   // VJS1499
   // try the following line instead
   // protected void Test()
   {
   }
   public static void main()
   {
   }
}
```

コンパイラの警告 (レベル 1) VJS1500

エラー メッセージ
型 '<型>' が 2 つ以上のインポートされたバイナリで見つかりました。'<ファイル名>' にあるものを使用します。

型が、複数の参照先アセンブリで定義されていました。Visual J# コンパイラは、<ファイル名> で定義された型を選択しました。

使用例

```
// VJS1500a.cs
// compile with: /target:library
// a C# program
public class MyClass
{
   public static int i = 1;
}
```

```
// VJS1500b.cpp
// compile with: /LD /clr
// a C++ program
public ref struct MyClass {
   static int i = 0;
};
```

```
// VJS1500c.jsl
// compile with: /reference:VJS1500a.dll /reference:VJS1500b.dll /W:1
// VJS1500 expected
public class MyClass2
{
   public static void main()
   {
      System.Console.WriteLine(MyClass.i);
   }
}
```

コンパイラの警告 (レベル 1) VJS1503

エラー メッセージ
'<メンバ>' は deprecated です。

deprecated とマークされたメンバが呼び出されました。

使用例

```
// VJS1503a.cs
// compile with: /target:library
// a C# program
public class VJS1503a
{
   [System.Obsolete ("deprecated")]
   public void Test() {}
}
```

次の例では、VJS1503 エラーが生成されます。

```
// VJS1503b.jsl
// compile with: /target:library /reference:VJS1503a.dll /W:1
class MyClass
{
   public void Test()
   {
      new VJS1503a () .Test ();   // VJS1503
   }
}
```

コンパイラの警告 (レベル 1) VJS1504

エラー メッセージ
パッケージ '<パッケージ名>' は既に暗黙的にインポートされています。

この警告は、コンパイラによって暗黙にインポートされるパッケージのインポートを試みた場合に表示されます。この警告を解決するには、明示的な **import** ステートメントを削除します。

次の例では、VJS1504 エラーが生成されます。

```
// VJS1504.jsl
// compile with: /target:library /W:1
import java.lang.*;  // VJS1504 Package java.lang is implicitly imported
class MyClass
{
}
```

コンパイラの警告 (レベル 1) VJS1505

エラー メッセージ
パッケージ '<パッケージ名>' は既にインポートされています。

パッケージ (または名前空間) が、2 回以上インポートされました。

次の例では、VJS1505 エラーが生成されます。

```
// VJS1505.jsl
// compile with: /W:1
import System.*;   // VJS1505
import System.*;
class Test
{
   public static void main(String[] args)
   {
   }
}
```

コンパイラの警告 (レベル 4) VJS1506

エラー メッセージ
'l' 接尾辞は、数字の '1' と混同される可能性があります。'L' を使用してください。

long 値を指定するときは、構文エラーの発生を避けるために、小文字の "l" ではなく大文字の "L" を使ってください。

次の例では、VJS1506 エラーが生成されます。

```
// VJS1506.jsl
// compile with: /W:4
class MyClass
{
   public static void TestL(long i)
   {
   }

   public static void TestL(int i)
   {
   }

   public static void main()
   {
      TestL(25l);   // VJS1506
      // try the following line instead
      // TestL(25L);
   }
}
```

コンパイラの警告 (レベル 3) VJS1507

エラー メッセージ
この 'instanceof' 演算子は常に true に設定されます。

instanceof チェックは常に成功し、true を返します。この状態は、**instanceof** 演算子を使って、派生クラスのオブジェクトが基本クラスのインスタンスと実装されたインターフェイスのインスタンスのどちらであるかを確認した場合に発生します。この警告は、チェックが冗長であるために表示されます。

次の例では、VJS1507 エラーが生成されます。

```
// VJS1507.jsl
// compile with: /W:3
interface I { }

class A { }

class B extends A implements I { }

public class MyClass
{
   public static void main (String [] args)
   {
      B b = new B();

      // Each of the instanceof operators below give VJS1507

      if (b instanceof B)   // VJS1507 b is of type B
      {
      }

      if (b instanceof A)   // VJS1507 b is instance of B that extends A
      {
      }

      if (b instanceof I)  // VJS1507 b is instance of B that implements I
      {
      }

      if (b instanceof Object)   // VJS1507 all classes derive from Object
      {
      }
   }
}
```

コンパイラの警告 (レベル 1) VJS1510

エラー メッセージ
空の switch ブロックです。

空の **switch** ブロックが検出されました。

次の例では、VJS1510 エラーが生成されます。

```
// VJS1510.jsl
// compile with: /W:1
class MyClass
{
   public static void main()
   {
      int i = 6;
      switch(i)
      {
         // to resolve, add something to the switch block, for example:
         /*
         case (5):
            System.Console.WriteLine("5");
         return;

         default:
            System.Console.WriteLine("not 5");
         return;
         */
      }   // VJS1510
   }
}
```

コンパイラの警告 (レベル 3) VJS1512

エラー メッセージ
変数 '<変数名>' は初期化されていますが、まだ使用されていません。

変数が宣言および初期化されましたが、使用されていません。

次の例では、VJS1512 エラーが生成されます。

```
// VJS1512.jsl
// compile with: /W:3
class Test
{
   public static void main(String[] args)
   {
      int i = 0;   // VJS1512
   }
}
```

コンパイラの警告 (レベル 1) VJS1513

エラー メッセージ
型 '<型>' が、インポートされたアセンブリおよびソースに見つかりました。ソースにある型を使用してください。

既存のアセンブリにある型が、ソース ファイルでも宣言されています。ソース ファイルの型が使用されます。

次の例では、VJS1513 エラーが生成されます。

```
// VJS1513.jsl
// compile with: /target:library /W:1
package java.util;
class Vector { }   // VJS1513
```

コンパイラ エラー VJS1515

エラー メッセージ
オプション 'オプション' はソースで指定された属性をオーバーライドします。

このエラーは、ソース ファイルに指定された属性が、コマンド ライン オプションと競合している場合に発生します。

このエラーを解決するには

- ソース ファイルから該当する行を削除するか、競合しているオプションを指定から外します。

使用例

次の例では、VJS1515 エラーが生成されます。

```
// VJS1515.jsl
// compile with: /keyfile:mykey.snk
/** @assembly System.Reflection.AssemblyKeyName("myotherkey.snk") */
class MyClass
{
   public static void main()
   {
   }
}
```

参照

その他の技術情報

Visual J# コンパイラ

コンパイラの警告 (レベル 2) VJS1516

エラー メッセージ
as volatile volatile フィールド 'フィールド' への参照は volatile として扱われます

volatile フィールドが Out パラメータとして渡されています。

このエラーを解決するには

-

使用例
次の例では、VJS1516 エラーが生成されます。

```
public class VolatileField {

    public static volatile R js_volatilefield_js;

    public static void method1(/**@attribute System.Runtime.InteropServices.OutAttribute()*
/ /**@ref*/ R myR) {

        myR = new R();

    }

    public static void method2() {

        js_volatilefield_js = new R();

        method1(js_volatilefield_js);

    }

}

class R{}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラの警告 (レベル 2) VJS1517

エラー メッセージ
型 long/double の volatile フィールド 'フィールド' は、volatile として扱われます

このエラーを解決するには

-

使用例
次の例では、VJS1517 エラーが生成されます。

```
public class Field {

static volatile double js_volatilefield_js = 0.001;

}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラの警告 (レベル 1) VJS1518

エラー メッセージ
保護されたスコープは、より広いスコープを含む基本クラス メソッドをオーバーライドするため、メンバ 'メンバ' 上で無視されます

このエラーを解決するには

- 安全なスコープ設定を本当にオーバーライドしてよいか確認します。

使用例

次の例では、VJS1518 エラーが生成されます。

```
public class Test
   {
    void foo()
      {
      }
   }
public class User extends Test
   {
    void foo()  // VJS1518 expected
      {
      }
   }
```

参照

その他の技術情報

Visual J# コンパイラ

コンパイラの警告 (レベル 3) VJS1519

エラー メッセージ
'MemberName': メンバ名がそれを囲む型と同じです

所属する型と同じ名前のメンバ名が指定されています。

このエラーを解決するには

- メンバ名を親の型と異なる名前に変更します。

使用例
次の例では、VJS1519 エラーが生成されます。

```
// VJS1519.jsl
// compile with: /t:l /w:3
public class VJS1519
{
    public void VJS1519a() { }
    //public void VJS1519(){}  //uncomment this line to generate VJS1519
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラの警告 (レベル 3) VJS1520

エラー メッセージ
Switch 文に 'case'、'default' がありません。

このエラーを解決するには

-

使用例
VJS1520 エラーを生成するには、default 句をコメント化します。

```
// VJS1520.jsl
// compile with:  /t:l /w:3

public class VJS1520
{
    int var = 0;
    public void method()
    {
        switch (var)
        {
            case 0:
                break;
            case 2:
                break;
            //default:   //uncomment out next two lines correct VJS1520
            //    System.out.println("invalid entry");
        }
    }
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラの警告 (レベル 3) VJS1522

エラー メッセージ
空の control ステートメントです

制御ステートメントが不完全です。

このエラーを解決するには

- 制御ステートメントに条件を追加します。

使用例
次の例では、VJS1522 エラーが生成されます。

```
// VJS1522.jsl
// compile with: /t:1 /w:3
public class VJS1522
{
    public void method()
    {
        if (true)
            System.out.println("Item is true"); //comment this line to generate VJS1522
        ;
    }
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラの警告 (レベル 2) VJS1523

エラー メッセージ
無効な #pragma チェックサム構文です。

#pragma チェックサム ディレクティブには、有効なファイル名、ファイルの GUID、チェックサムのバイトという 3 つの要素が含まれている必要があります。詳細については、「#pragma checksum (Visual J#)」を参照してください。

このエラーを解決するには

- ディレクティブの 3 つの要素がすべて存在することを確認します。

使用例
次の例では、チェックサムのバイトに値が存在しないため、VJS1523 が生成されます。

```
#pragma checksum    "drt1.java"   "{BED7F4EA-1A96-11d2-8F08-00A0C9A6186D}"     ""
// VJS1523 expected
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1524

エラー メッセージ
@dll.import ディレクティブは DLL 名を指定しなければなりません。

不正な形式の **@dll.import** 指定が検出されました。

次の例では、VJS1524 エラーが生成されます。

```
// VJS1524.jsl
class MyClass
{
   /**
   * @dll.import()   // VJS1524, no .dll file specified
   // try the following line instead
   // * @dll.import("msvcrt", auto)
   */
   public static native int puts(System.String c);

   public static void main()
   {
   }
}
```

コンパイラの警告 (レベル 3) VJS1525

エラー メッセージ
パラメータ名 '<パラメータ名 1>' は、対応する @com.parameters パラメータ名 '<パラメータ名 2>' と一致しません。

メソッド宣言のパラメータ名と **@com** 属性で指定されたパラメータ名は、一致する必要があります。

次の例では、VJS1525 エラーが生成されます。

```
// VJS1525.jsl
// compile with: /reference:TESTCOMOBJLib.dll /target:library /W:3
package testcomobj;

/** @com.class(classid=836A4F1C-AF14-4210-99DB-28C7BDD2DE2B,DynamicCasts) */
public class Calculator   // contained in TEstCOMObjLib.dll
{
   /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add", addFlagsVtable=4)
       @com.parameters([in,type=I4] a, [in,type=I4] b, [type=I4] return) */   // VJS1525
   public native int Add(int i, int b);
   // try the following line instead
   // public native int Add(int a, int b);

}
```

コンパイラ エラー VJS1526

エラー メッセージ
'return' は @com.parameters 宣言で最後でなければなりません。

@com.parameters 宣言で、**return** ステートメントの後に識別子が検出されました。

次の例では、VJS1526 エラーが生成されます。

```
// VJS1526.jsl
// compile with: /target:library
class MyClass
{
   /** @com.parameters([type=I4] return, i) */   // VJS1526
   // try the following line instead
   // /** @com.parameters([type=I4] i, return) */
   native int method1(Object i);
}
```

コンパイラ エラー VJS1527

エラー メッセージ
'return' は @com.parameters で既に定義されています。

@com.parameters 宣言で、**return** ステートメントが重複しています。

次の例では、VJS1527 エラーが生成されます。

```
// VJS1527.jsl
// compile with: /target:library
class MyClass
{
   /** @com.parameters([type=I4] i, return, return) */   // VJS1527
   // try the following line instead
   // /** @com.parameters([type=I4] i, return) */
   native int method1(int i);
}
```

コンパイラ エラー VJS1528

エラー メッセージ
@com.parameters ディレクティブには、メンバ '<メンバ名>' に対して間違ったパラメータ番号が指定されています。

メソッド宣言のパラメータ数と **@com** 属性で指定されたパラメータ数は、一致する必要があります。

次の例では、VJS1528 エラーが生成されます。

```
// VJS1528.jsl
// compile with: /target:library /reference:testcomobjlib.dll
// assumes the presence of a COM object, testcomobjlib.dll
/** @com.class(classid=836A4F1C-AF14-4210-99DB-28C7BDD2DE2B, DynamicCasts) */
public class Calculator
{
   /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add", addFlagsVtable=4)
       @com.parameters([in,type=I4] a, [type=I4] return) */
   public native int Add(int a, int b);   // VJS1528
   // try the following line instead
   // public native int Add(int a);
}
```

コンパイラ エラー VJS1529

エラー メッセージ
'<パラメータ名>' が '@<属性>' に必要です。

属性の定義が不完全です。

次の例では、VJS1529 エラーが生成されます。

```
// VJS1529.jsl
// compile with: /target:library
/** @com.class() */   // VJS1529
// try the following line instead
// /** @com.class(classid=911CAED0-2957-11d1-A55E-00A0C90F26EE) */
public class Calculator
{
}
```

コンパイラ エラー VJS1530

エラー メッセージ
'@<属性>' はこの宣言に対して有効ではありません。

属性が、意図していない構成要素に適用されました。

次の例では、VJS1530 エラーが生成されます。

```
// VJS1530.jsl
/** @com.register (clsid=2148925C-234F-11D2-8C4E-00C04F8F3341, typelib=2148925B-234F-11D2-8
C4E-00C04F8F3341) */   // VJS1530
interface MyInterface
{
}
```

コンパイラ エラー VJS1531

エラー メッセージ
GUID '<GUID>' をもつ .NET COM クラスが見つかりません。

クラスへの参照が解決されませんでした。

次の例では、VJS1531 エラーが生成されます。

```
// VJS1531.jsl
// compile with: /target:library
/** @com.class(classid=911CAED0-2957-11d1-A55E-00A0C90F26EE) */   // VJS1531
// To resolve, reference a COM object on which JActiveX or uuidgen was run
// and that has a class with the specified ID.
class MyClass
{
   /** @com.parameters([type=I4], return) */
   int method1(int i);
}
```

コンパイラ エラー VJS1532

エラー メッセージ
GUID '<GUID>' をもつ .NET COM インターフェイスが見つかりません。

インターフェイスへの参照が解決されませんでした。

次の例では、VJS1532 エラーが生成されます。

```
// VJS1532.jsl
// compile with: /target:library
/** @com.interface(iid=911CAED0-2957-11d1-A55E-00A0C90F26EE) */   // VJS1532
// to resolve, reference a COM object on which JActiveX or uuidgen was run
// and that has an interface with the specified ID
interface MyInterface
{
}
```

コンパイラ エラー VJS1533

エラー メッセージ
メソッド '<メソッド名>' と一致する .NET メソッドが、COM クラス/インターフェイス '<型>' で見つかりません。

JActiveX ツールで生成された属性付きインターフェイス ラッパーをコンパイルするには、タイプ ライブラリで、TlbImp が生成した DLL を参照する必要があります。Visual J# コンパイラは、インターフェイスの各メソッドに一致するメソッドを、TlbImp が生成した DLL の対応するインターフェイスで探します。以下に示すサンプルでは、**@com.interface** 属性付きインターフェイスのメソッドの名前およびシグネチャが一致するメソッドが見つかりませんでした。

`Add1` は、`ICalculator` インターフェイスのメソッドではないため、エラーが生成されます。

次の例では、VJS1533 エラーが生成されます。

```
// VJS1533.jsl
// compile with: /target:library /reference:TESTCOMOBJLib.dll
// assumes the presence of a COM object, testcomobjlib.dll
import com.ms.com.IUnknown;

// Dual interface ICalculator
/** @com.interface(iid=7F707219-A772-4437-AD86-DBABD5F6BC33, thread=AUTO, type=DUAL) */
public interface ICalculator extends IUnknown
{
   /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add1", addFlagsVtable=4)
       @com.parameters([in,type=I4] a, [in,type=I4] b, [type=I4] return) */
   public int Add1(int f, int b);   // VJS1533
}
```

コンパイラ エラー VJS1534

エラー メッセージ
パラメータ '<パラメータ名>' に対応する @com.parameters ディレクティブは 'type' の値を指定していません。

@com.parameters ディレクティブにパラメータの型属性がありません。

次の例では、VJS1534 エラーが生成されます。

```
// VJS1534.jsl
// compile with: /target:library /reference:TESTCOMOBJLib.dll
// assumes the presence of a COM object, testcomobjlib.dll
/** @com.interface(iid=7F707219-A772-4437-AD86-DBABD5F6BC33, thread=AUTO, type=DUAL) */
public interface ICalculator extends com.ms.com.IUnknown
{
   /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add", addFlagsVtable=4)
       @com.parameters([in] a, [in,type=I4] b, [type=I4] return) */   // VJS1534
       // try the following line instead
       // @com.parameters([in, type=I4] a, [in,type=I4] b, [type=I4] return) */
   public int Add(int a, int b);
}
```

コンパイラ エラー VJS1535

エラー メッセージ
パラメータ '<パラメータ名>' の @com.parameters ディレクティブには、サポートされていない型が指定されています。

@com.parameters ディレクティブの型値は STRUCT または PTR ですが、実パラメータは STRUCT ではありません。以下に示すサンプルで、パラメータ `a` は `STRUCT` 型 (**@com.struct** 属性付きクラス) ではなく、`Object` 型です。

次の例では、VJS1535 エラーが生成されます。

```
// VJS1535.jsl
// compile with: /target:library /reference:TESTCOMOBJLib.dll
// assumes the presence of a COM object, testcomobjlib.dll
import com.ms.com.IUnknown;
// Dual interface ICalculator
/** @com.interface(iid=7F707219-A772-4437-AD86-DBABD5F6BC33, thread=AUTO, type=DUAL) */
interface ICalculator extends com.ms.com.IUnknown
{
   /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add", addFlagsVtable=4)
       @com.parameters([in,type=STRUCT] a, [in,type=I4] b, [type=I4] return) */   // VJS153
5
   public int Add(Object a, int b);
}
```

コンパイラ エラー VJS1537

エラー メッセージ
パラメータ '<パラメータ名>' の @com.parameters ディレクティブには、サポートされていない SAFEARRAY VT 型が指定されています。

@com.parameters ディレクティブのパラメータは SAFEARRAY (**type=SAFEARRAY**) ですが、**vt** 属性の値がサポートされていません。

Visual J# でサポートされているのは、次の値だけです。

- VT_I2 = 2
- VT_I4 = 3
- VT_R4 = 4
- VT_R8 = 5
- VT_CY = 6
- VT_DATE = 7
- VT_BSTR = 8
- VT_DISPATCH = 9
- VT_BOOL = 11
- VT_VARIANT = 12
- VT_UNKNOWN = 13
- VT_I1 = 16
- VT_UI1 = 17

次の例では、VJS1537 エラーが生成されます。

```
// VJS1537.jsl
// compile with: /target:library /reference:TESTCOMOBJLib.dll
// assumes the presence of a COM object, testcomobjlib.dll
import com.ms.com.IUnknown;
// Dual interface ICalculator
/** @com.interface(iid=7F707219-A772-4437-AD86-DBABD5F6BC33, thread=AUTO, type=DUAL) */
public interface ICalculator extends IUnknown
{
   /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add", addFlagsVtable=4)
       @com.parameters([in,type=SAFEARRAY,vt=10] a, [in,type=I4] b, [type=I4] return) */
// VJS1537
   public int Add(com.ms.com.SafeArray a, int b);
   // don't use Add or change its definition in the COM object
   // to use a different VT type
}
```

コンパイラ エラー VJS1538

エラー メッセージ
パラメータ '<パラメータ名>' に対応する @com.parameters ディレクティブは、サポートされていないカスタム マーシャリングを使用します。

@com.parameters ディレクティブに、カスタム マーシャリングの使用 (**type=CUSTOM**) が指定されています。vjprvj は、@com.parameters ディレクティブのカスタム マーシャリングの使用をサポートしていません。

次の例では、VJS1538 エラーが生成されます。

```
// VJS1538.jsl
// compile with: /target:library /reference:TESTCOMOBJLib.dll
// assumes the presence of a COM object, testcomobjlib.dll
import com.ms.com.IUnknown;

// Dual interface ICalculator
/** @com.interface(iid=7F707219-A772-4437-AD86-DBABD5F6BC33, thread=AUTO, type=DUAL) */
public interface ICalculator extends IUnknown
{
   /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add", addFlagsVtable=4)
       @com.parameters([in,type=CUSTOM,customMarshal="myMarshallerClass"] a, [in,type=I4] b
, [type=I4] return) */   // VJS1538
   public int Add(int a, int b);
}
```

コンパイラ エラー VJS1539

エラー メッセージ
COM インターフェイス ラッパー用の中間ソース ファイルを作成できません。

@com 属性付きの Java 言語ソース ファイルのコンパイル時には、コンパイラによって一時ファイルが作成されます。このエラーは、コンパイラが一時ファイルを作成できない場合に発生します。一時ファイルは、ユーザーの一時ディレクトリか現在のディレクトリに作成されます。

このエラーは、一時ディレクトリ (または現在のディレクトリ) への書き込みのアクセス許可がない場合か、ディスクがいっぱいである場合に発生します。

コンパイラ エラー VJS1540

エラー メッセージ
@dll または @com において、'<パラメータ名>' は '<クラス名>' に対して無効です。

属性で無効なパラメータが見つかりました。

次の例では、VJS1540 エラーが生成されます。

```
// VJS1540.jsl
// compile with: /target:library
/** @dll.import("test.dll", entrypoint="Test") */   // VJS1540
// try the following line instead
// /** @dll.import("test.dll") */
public class MyClass
{
}
```

コンパイラ エラー VJS1541

エラー メッセージ
@dll または @com では、'<メンバ名>' は '<パラメータ名>' を必要とします。

情報が不足しているため、属性の形式が不正です。

次の例では、VJS1541 エラーが生成されます。

```
// VJS1541.jsl
// compile with: /target:library
// VJS1541 expected
/** @dll.struct(noAutoOffset) */
public class MyClass
{
   /** @dll.structmap() */
   // Try the following line instead:
   // /** @dll.structmap([offset=0]) */
   int i;
}
```

コンパイラ エラー VJS1542

エラー メッセージ
'@<属性名>' に対して 'pack' および 'noAutoOffset' を同時に指定することはできません。

同時に指定できないパラメータが属性に指定されています。

次の例では、VJS1542 エラーが生成されます。

```
// VJS1542.jsl
// compile with: /target:library
/** @dll.struct(noAutoOffset, pack=1) */   // VJS1542
// try the following line instead
// /** @dll.struct(noAutoOffset) */
public class MyClass
{
}
```

コンパイラ エラー VJS1543

エラー メッセージ
GUID 宣言の構文エラーが '@<属性名>' にあります。

GUID の指定が無効です。

次の例では、VJS1543 エラーが生成されます。

```
// VJS1543.jsl
// compile with: /target:library
/** @com.class(classid=x) */   // VJS1543, add real classid
public class MyClass
{
}
```

コンパイラ エラー VJS1544

エラー メッセージ
'@<属性>' 宣言に構文エラーがあります。

属性宣言で構文エラーが見つかりました。

次の例では、VJS1544 エラーが生成されます。

```
// VJS1544.jsl
class MyClass
{
   /**
      * @dll.import(,auto)   // VJS1544, no .dll file specified
      // try the following line instead
      // * @dll.import("msvcrt",auto)
    */
    public static native int puts(System.String c);

   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1545

エラー メッセージ
@dll.import は '<メソッド名>' が native であることを要求します。

@dll.import で宣言されたメソッドは、**native** 修飾子を使って宣言する必要があります。

次の例では、VJS1545 エラーが生成されます。

```
// VJS1545.jsl
// compile with: /target:library
class MyClass
{
   /**
   * @dll.import("msvcrt", auto)
   */
   public static int puts(System.String c);   // VJS1545
   // try the following line instead
   // public static native int puts(System.String c);

   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1546

エラー メッセージ
クラス/インターフェイス '<型>' に対応する .NET 型が、重複するパッケージ名を使っています。tlbimp.exe の /namespace または /out オプションを使用して、タイプ ライブラリを他のパッケージにインポートしてください。

@com.interface 属性付きインターフェイスのパッケージ名が、TlbImp.exe が生成したクラスのパッケージ名と同じです。

次に示すサンプルでは、TlbImp が生成した DLL である TESTCOMOBJLib.dll の .NET Framework クラスのパッケージ名とも同じです。このエラーを解決するには、`tlbimp /namespace=SomethingElse TestComObj.dll` を使って再コンパイルします。

次の例では、VJS1546 エラーが生成されます。

```
// VJS1546.jsl
// compile with: /target:library /reference:TESTCOMOBJLib.dll
// assumes the presence of a COM object, testcomobjlib.dll
package TESTCOMOBJLib;
import com.ms.com.IUnknown;

// Dual interface ICalculator
/** @com.interface(iid=7F707219-A772-4437-AD86-DBABD5F6BC33, thread=AUTO, type=DUAL) */   /
/ VJS1546
public interface ICalculator extends IUnknown
{
  /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add", addFlagsVtable=4)
      @com.parameters([in,type=I4] a, [in,type=I4] b, [type=I4] return) */
 public int Add(int a, int b);
}
```

コンパイラ エラー VJS1547

エラー メッセージ
一致する .NET フィールド '<フィールド名>' がクラス '<型>' に見つかりません。これは J/Direct に必要です。

JActiveX ツールで生成された **@com** 属性付きインターフェイス ラッパーをコンパイルするには、TlbImp が生成した DLL でタイプ ライブラリを参照する必要があります。Visual J# コンパイラは、インターフェイスの各メソッドに一致するメソッドを、TlbImp が生成した DLL のインターフェイスで探します。次に示すサンプルでは、TlbImp が生成したクラスで一致するフィールドが見つかりませんでした。

このサンプルのエラーを解決するには、フィールド宣言を削除するか、`TESTCOMOBJLib.MyStruct` クラスのフィールド宣言のうちの 1 つに割り当てます。

次の例では、VJS1547 エラーが生成されます。

```
// VJS1547.jsl
// compile with: /target:library /reference:TESTCOMOBJLib.dll
// assumes the presence of a COM object, testcomobjlib.dll
package testcomobj;

/** @com.struct(noAutoOffset) */

public final class MyStruct   // VJS1547
{
   /** @com.structmap([offset=0,type=I4] fld1) */
   public int fld1;

   /** @com.structmap([offset=4,type=I4] fld2) */
   public int fld2;
}
```

コンパイラ エラー VJS1548

エラー メッセージ
.NET 型 '<型 2>' と競合する '<型 1>' の適切な tlbimported .NET 型をバインドする /jcpa:@<file_name> オプションを使用します。

2 つの異なるライブラリに、同じ COM インターフェイスのインスタンスが 2 つあります。また、対応する 2 つの .NET Framework インターフェイスも、異なる名前空間 (別個のアセンブリ) に存在します。

コンパイラは、COM インターフェイスと .NET Framework インターフェイスの対応を解決できません。/jcpa コンパイラ オプションを使って、対応を明示的に指定してください。

コンパイラ エラー VJS1549

エラー メッセージ
COM インターフェイス '<インターフェイス名>' のメソッド '<メソッド名>' の @com.parameters ディレクティブがありません。

@com 属性付き COM インターフェイスの COM メソッドに、**@com.parameters** ディレクティブがありません。

次の例では、VJS1549 エラーが生成されます。

```
// VJS1549.jsl
// compile with: /target:library /reference:TESTCOMOBJLib.dll
// assumes the presence of a COM object, testcomobjlib.dll
import com.ms.com.IUnknown;
/** @com.interface(iid=7F707219-A772-4437-AD86-DBABD5F6BC33, thread=AUTO, type=DUAL) */
interface ICalculator extends IUnknown
{
   /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add", addFlagsVtable=4) */
   // add the following line to the attribute block
   // @com.parameters([in,type=I4] a, [in,type=I4] b, [type=I4] return)
   public int Add(int a, int b);   // VJS1549
}
```

コンパイラ エラー VJS1550

エラー メッセージ
@com.parameters ディレクティブには、COM インターフェイス '<インターフェイス名>' のメソッド '<メソッド名>' のすべてのパラメータに関する情報が含まれていません。

@com.parameters ディレクティブに、一部のパラメータの記述が含まれていません。

次の例では、VJS1550 エラーが生成されます。

```
// VJS1550.jsl
// compile with: /target:library /r:TESTCOMOBJLib.dll
// assumes the presence of a COM object, testcomobjlib.dll
import com.ms.com.IUnknown;
/** @com.interface(iid=7F707219-A772-4437-AD86-DBABD5F6BC33, thread=AUTO, type=DUAL) */
interface ICalculator extends IUnknown
{
  /** @com.method(vtoffset=4, dispid=1, type=METHOD, name="Add", addFlagsVtable=4)
      @com.parameters([in,type=I4] a, [type=I4] return) */
      // try the following line instead
      // @com.parameters([in,type=I4] a, [in,type=I4] b, [type=I4] return) */
 public int Add(int a, int b);   // VJS1550

}
```

コンパイラの警告 (レベル 1) VJS1551

エラー メッセージ
'@<属性>' はここでは有効ではありません。無視されます。

属性が、意図していない宣言に適用されました。属性は無視されました。

次の例では、VJS1551 エラーが生成されます。

```
// VJS1551.jsl
// compile with: /target:library /W:1
class MyClass
{
   /** @dll.struct () */   // VJS1551
   int x;
}
```

コンパイラ エラー VJS1552

エラー メッセージ
'<値>' は、/jcpa に対して有効な値ではありません。

無効な値が /jcpa コンパイラ オプションに渡されました。

次の例では、VJS1552 エラーが生成されます。

```
// VJS1552.jsl
// compile with: /jcpa:#
// VJS1552 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラの警告 (レベル 2) VJS1553

エラー メッセージ
クラス '<comclass>' の COM タイプ ライブラリが見つかりませんでした。タイプ ライブラリを使用してこのクラスにアクセスしていた COM アプリケーションは、使用できなくなります。

@com.register 属性でタイプ ライブラリの GUID が指定されていないか、指定された GUID がコンピュータに登録されていません。

Visual J++ 6.0 では、**@com.register** 属性でマークされた Java 言語/COM コンポーネントに COM クライアントからアクセスできます。VJREG ツールがコンポーネントで生成したタイプ ライブラリを使って COM クライアントが Java 言語/COM コンポーネントにアクセスする場合、COM クライアントはコンポーネントで公開されたディスパッチ インターフェイスを使い、メンバの DISPID をキャッシュに保存します。この種のコンポーネントの DISPID が .NET Framework でも同一であることは保証されません。このため、既存の COM クライアントで問題が発生する場合があります。この実行時エラーを避けるには、COM クライアントが使っていたのと同じ DISPID をコンパイラで生成する必要があります。この警告は、コンパイラがタイプ ライブラリを読み込むことができないために、ディスパッチ インターフェイスを通じてオブジェクトを使う COM クライアントが実行時に正しく動作し続けることが保証されない場合に表示されます。

この問題に対処するには、次の操作を行います。

- **typelib** 属性が指定されていない場合は、**@com.register** クラスに追加します。**typelib** の GUID 値は、VJREG ツールが生成するタイプ ライブラリに割り当てられる GUID と同じである必要があります。
- 指定した GUID を持つタイプ ライブラリが、コンピュータに登録されていることを確認します。

次の例では、VJS1553 エラーが生成されます。

```
// VJS1553.jsl
// compile with: /target:library /W:2
// typelib attribute missing in @com.register directive
/** @com.register(clsid=b62107bb-18ed-4796-a61a-0ade6edd3a91) */
public class test{}   // VJS1553
// typelib attribute found but type library is not registered on machine
/** @com.register(clsid=b62107bb-18ed-4796-a61a-0ade6edd3a91, typelib=84b1fa15-6a14-4663-be
9e-aa23fe40090d) */
public class test2{}   // VJS1553
```

コンパイラ エラー VJS1554

エラー メッセージ
DllImportAttribute を指定するには、メソッド '<メソッド名>' が static および native でなければなりません。

DLLImportAttribute は、static または native なメソッドにだけ追加できます。

次の例では、VJS1554 エラーが生成されます。

```
// VJS1554.jsl
import System.Runtime.InteropServices.*;

public class MyClass {
   /** @attribute DllImportAttribute("Test") */
   public static void Compute();   // VJS1554
   // try the following line instead
   // public static native void Compute();
}
```

コンパイラ エラー VJS1555

エラー メッセージ
クラス '<クラス名>' は、パブリック クラスでないため、COM に公開できません。

@com.register 属性を使って、COM に公開できるのはパブリック クラスだけです。

このエラーは、クラスで COM インターフェイスを実装し、入れ子になったプライベート クラスを COM に公開する場合に表示されます。

使用例

```
// VJS1555a.jsl
// compile with: /target:library
/** @com.register(clsid=17d49bf0-d80f-4cb4-a40f-a17b1037bee5) */
// class MyClass {}   // VJS1555
// try the following line instead
// public class MyClass {}
```

```
// VJS1555b.jsl
// compile with: /target:library
import com.ms.com.*;
public class MyOuterClass
{
   private class MyEnum implements IEnumVariant   // VJS1555
   // try the following line instead
   // public class MyEnum implements IEnumVariant
   {
      public IEnumVariant Clone() { return null; }
      public void Next(int celt, Variant[] rgvar, int[] pceltFetched) {}
      public void Reset() {}
      public void Skip(int celt) {}
   }
}
```

コンパイラ エラー VJS1561

エラー メッセージ
'<ファイル名>' : ファイル形式が win32 リソース ファイル形式と一致しません。

/win32res コンパイラ オプションに渡されたファイルは、Win32 リソース ファイル形式ではありません。

このエラーを解決するには

- コンパイラに渡そうとしているファイルが Win32 のリソース ファイルであることを確認します。

使用例
次の例では、VJS1561 エラーが生成されます。

```
// VJS1561.jsl
// compile with: /win32res:VJS1561.doc
// VJS1561 expected
// assume VJS1561.doc is a file on disk
public class MyClass
{
   public static void main()
   {
   }
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1562

エラー メッセージ
ファイルが見つからないか、読み取れません : '<ファイル名>'

/win32res、/resource、または /linkresource に渡されたファイルが見つからないか、または読み取ることができません。

次の例では、VJS1562 エラーが生成されます。

```
// VJS1562.jsl
// compile with: /win32res:x.res
// VJS1562 expected
// assume x.res is not on disk
public class MyClass
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS1566

エラー メッセージ
win32 リソース内の 2 つのリソース データ エントリは同じ種類、同じ名前、同じ言語です。このようなリソースをアタッチすることはできません。

/win32res で指定されたリソース ファイル (.res) は破損しています。ファイル形式が正しいにもかかわらず、不正なデータがあります。

たとえば、同じリソースを .res ファイルに 2 回追加した場合や、.res ファイルを自分自身に追加してから **/win32res** オプションを使ってそのファイルを追加しようとした場合は、このエラーが生成されます。

このエラーを解決するには

1. リソース ファイルを探します。
2. ファイルが破損していないか確認します。
3. コードをチェックし、リソース ファイルが 2 回以上追加されていないか確認します。

使用例
次の例では、VJS1566 エラーが生成されます。

```
//steps to reproduce:
//copy some.res VJS1566.res
//type some.res >> VJS1566.res
//vjc /win32res:VJS1566.res test.java

// VJS1566.jsl
// compile with: /win32res:VJS1566.res
// VJS1566 expected
public class MyClass
{
   public static void main()
   {
   }
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1567

エラー メッセージ
競合するオプションが指定されました : Win32 リソース ファイル、Win32 アイコン

このエラーを解決するには

-

使用例
次の例では、VJS1567 エラーが生成されます。

```
// VJS1567.jsl
// compile with: /win32res:MSN.ico /win32res:cmredz67.res
package ConsoleApplication1;

/**
 * Summary description for Program
 */
//
public class Program
{
public static void main(String[] args)
{
}
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1568

エラー メッセージ
Win32 リソースの生成中にエラーが発生しました: アイコン 'アイコン' の読み取りエラー-- データが無効です

無効なアイコン ファイルが /win32icon に指定されています。

このエラーを解決するには

- アイコン データを確認して、やり直します。

使用例
次の例では、VJS1568 エラーが生成されます。

```
// compile with /win32icon:cmredz67.res sample.java //VJS1568 expected
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1575

エラー メッセージ
直接または間接を問わず、コンストラクタを再帰的に呼び出すことができません。

このエラーは、コンストラクタの内部でコンストラクタの再帰呼び出しを試みたときに発生します。

次の例では、VJS1575 エラーが生成されます。

```
// VJS1575.jsl
// compile with: /target:library
public class MyClass
{
   MyClass()
   {
      this();   // VJS1575, recursive call to the same constructor
   }
}
```

コンパイラ エラー VJS1576

エラー メッセージ
コンパイラ中間ファイルのディレクトリを作成できません。

コンパイラは、コンパイル時に一時ディレクトリを作成します。このエラーは、コンパイラが一時ディレクトリを作成できない場合に発生します。

つまり、ディレクトリへの書き込みのアクセス許可がない場合か、ディスクがいっぱいである場合です。

コンパイラ エラー VJS1578

エラー メッセージ
Visual J# システム ディレクトリを取得できません。

このエラーは、Visual J# インストールが破損している場合に発生します。ディレクトリの一部が削除されたり、レジストリが破損したりする可能性があります。

このエラーを解決するには、Visual J# の修復または再インストールを実行します。

コンパイラ エラー VJS1579

エラー メッセージ
ユーザーがビルドをキャンセルしました。

ビルド処理中に、開発環境で [ビルド] メニューの [キャンセル] がクリックされました。

コンパイラ エラー VJS1580

エラー メッセージ
内部コンパイラ エラー - メモリが足りません。

ソースのコンパイル中に、使用できるメモリがなくなりました。次のどちらかの操作を実行してから、コンパイルを再試行してください。

- 実行中の 1 つ以上のアプリケーションを終了して、使用できるメモリを確保します。
- 仮想メモリのページファイル サイズを増やします。詳細については、オペレーティング システムのオンライン ヘルプを参照するか、システム管理者に連絡してください。

コンパイラ エラー VJS1581

エラー メッセージ
内部コンパイル エラー : <理由>

コンパイラで予測不可能な構文を解析できないためにエラーが発生したのかどうかを調べてください。次に、マイクロソフト製品サポート サービスに問い合わせます。

コンパイラ エラー VJS1582

エラー メッセージ
属性 'type' をアタッチ中にエラーが発生しました - 'メッセージ'

特定のメンバに属性を追加できませんでした。

次の例では、VJS1582 エラーが生成されます。

```
// VJS1582.jsl
// compile with: /target:library
// VJS1582 expected
/** @attribute System.Runtime.InteropServices.GuidAttribute("") */
class MyClass
{
}
```

コンパイラ エラー VJS1800

エラー メッセージ
プロパティ アクセサは、get_ または set_ で始めなければなりません。

プロパティ名の形式が不正です。

次の例では、VJS1800 エラーが生成されます。

```
// vjs1800.jsl
// compile with: /target:library
public class C
{
   /** @property */
   int Val()   // VJS1800
   // try the following line instead
   // int get_Val()
   {
      return 0;
   }
}
```

コンパイラ エラー VJS1801

エラー メッセージ
プロパティ名 '<名前>' は、有効な識別子ではありません。

定義されたプロパティ名がクライアントで使用されるときは、プリフィックス get_ または set_ を除いた名前に解決されます。識別子の先頭に無効な特定の文字があります。プロパティ名のプリフィックス get_ または set_ の後の最初の文字は、識別子の有効な開始文字であることが必要です。たとえば、プロパティ名の開始文字として数字は使用できません。

次の例では、VJS1801 エラーが生成されます。

```
// VJS1801.jsl
// compile with: /target:library
public class MyClass
{
   /** @property */
   int get_2Val()   // VJS1801
   // try the following line instead
   // int get_Val()
   {
      return 0;
   }
}
```

コンパイラ エラー VJS1802

エラー メッセージ
'<識別子>' は、プロパティまたはイベント名として再定義されています。

Visual J# ソースに定義されたプロパティ名をユーザーが使うときは、get_ または set_ を除きます。つまり、コンポーネントのコンシューマが使うプロパティ名と同じ名前の識別子は定義できません。

次の例では、VJS1802 エラーが生成されます。

```
// VJS1802.jsl
// compile with: /target:library
public class MyProp
{
   private int SomeNum = -1;
   // try the following line instead
   // private int i = -1;

   /** @property */
   public int get_SomeNum()   // VJS1802, SomeNum already defined
   {
      return i;
   }
}
```

コンパイラ エラー VJS1803

エラー メッセージ
プロパティの型を void にすることはできません。

プロパティの get_ アクセサは、値を返す必要があります。

次の例では、VJS1803 エラーが生成されます。

```
// VJS1803.jsl
// compile with: /target:library
public class MyClass
{
   private int ival;

   /** @property */
   public void get_Val()   // VJS1803
   // try the following line instead
   // public int get_Val()
   {
      return 0;
   }
}
```

コンパイラ エラー VJS1804

エラー メッセージ
set アクセサ メソッドには、void の戻り値の型が必要です。

プロパティの set アクセサの戻り値の型が void ではありませんでした。

次の例では、VJS1804 エラーが生成されます。

```
// VJS1804.jsl
// compile with: /target:library
public class MyClass
{
   private int ival;

   /** @property */
   public int set_Val(int i)   // VJS1804
   // try the following line instead
   // public void set_Val(int i)
   {
      ival = i;
   }
}
```

コンパイラ エラー VJS1805

エラー メッセージ
プロパティまたはイベントの両方のアクセサ メソッドには、同じ修飾子を指定しなければなりません。

プロパティまたはイベントのアクセサ メソッドには、同じアクセス修飾子を使う必要があります。

次の例では、VJS1805 エラーが生成されます。

```
// vjs1805.jsl
// compile with: /target:library
public class MyClass
{
   private int ival;

   /** @property */
   public int get_Val()
   {
      return ival;
   }

   /** @property */
   private void set_Val(int i)   // VJS1805
   // try the following line instead
   // public void set_Val(int i)
   {
      ival = i;
   }
}
```

コンパイラ エラー VJS1806

エラー メッセージ
イベント '<イベント名>' には、add および remove アクセサの両方を指定しなければなりません。

イベントの定義が不完全です。

次の例では、VJS1806 エラーが生成されます。

```
// vjs1806.jsl
// compile with: /target:library
/** @delegate */
public delegate void ValueChanged();

public class Container   // VJS1806
{
   public ValueChanged ev = null;

   /** @event */
   public void add_Event(ValueChanged p)
   {
      ev = (ValueChanged) System.Delegate.Combine(ev, p);
   }

   // event with add accessor also needs remove accessor
   // /** @event */
   // public void remove_Event(ValueChanged p)
   // {
   //    ev = (ValueChanged) System.Delegate.Remove(ev, p);
   // }
}
```

コンパイラ エラー VJS1807

エラー メッセージ
イベント名 '<識別子>' は、有効な識別子ではありません。

定義されたイベント名がクライアントで使用されるときは、プリフィックス **add_** または **remove_** を除いた名前に解決されます。識別子の先頭に無効な特定の文字があります。イベント名のプリフィックス **add_** または **remove_** の後の最初の文字は、識別子の有効な開始文字であることが必要です。たとえば、イベント名の開始文字として数字は使用できません。

次の例では、VJS1807 エラーが生成されます。

```
// vjs1807.jsl
// compile with: /target:library
/** @delegate */
public delegate void ValueChanged();

public class Container
{
   public ValueChanged ev = null;

   /** @event */
   public void add_2Event(ValueChanged p)   // VJS1807
   // try the following line instead
   // public void add_Event(ValueChanged p)
   {
      ev = (ValueChanged) System.Delegate.Combine(ev, p);
   }

   /** @event */
   public void remove_2Event(ValueChanged p)  // VJS1807
   {
      ev = (ValueChanged) System.Delegate.Remove(ev, p);
   }
}
```

コンパイラ エラー VJS1808

エラー メッセージ
イベントの型 '<識別子>' は、delegate ではありません。

識別子は、デリゲートとして宣言されていませんが、デリゲートとして使用されました。

次の例では、VJS1808 エラーが生成されます。

```
// vjs1808.jsl
// compile with: /target:library
// uncomment the following line to resolve
// /** @delegate */
public delegate void ValueChanged();

public class Container
{
   public ValueChanged ev = null;

   /** @event */
   public void add_Event(ValueChanged p)   // VJS1808
   {
      ev = (ValueChanged) System.Delegate.Combine(ev, p);
   }

   /** @event */
   public void remove_Event(ValueChanged p)
   {
      ev = (ValueChanged) System.Delegate.Remove(ev, p);
   }
}
```

コンパイラ エラー VJS1809

エラー メッセージ
Event アクセサには、void の戻り値の型を指定しなければなりません。

イベント アクセサの戻り値の型が無効です。

次の例では、VJS1809 エラーが生成されます。

```
// vjs1809.jsl
// compile with: /target:library
/** @delegate  */
public delegate void ValueChanged();

public class Container
{
   public ValueChanged ev = null;

   /** @event */
   public int add_Event(ValueChanged p)   // VJS1809
   // try the following line instead
   // and delete the return statement from the body of the event
   // public void add_Event(ValueChanged p)
   {
      ev = (ValueChanged) System.Delegate.Combine(ev, p);
      return 0;
   }

   /** @event */
   public void remove_Event(ValueChanged p)
   {
      ev = (ValueChanged) System.Delegate.Remove(ev, p);
   }
}
```

コンパイラ エラー VJS1810

エラー メッセージ
Event アクセサの型は、'void add_name(type)' または 'void remove_name(type)' でなければなりません。'type' は、System.Delegate から派生します。

Event アクセサの引数の数が無効です。

次の例では、VJS1810 エラーが生成されます。

```
// vjs1810.jsl
// compile with: /target:library
/** @delegate  */
public delegate void ValueChanged();

public class Container
{
   public ValueChanged ev = null;

   /** @event */
   public void add_Event(ValueChanged p, int i)   // VJS1810
   // try the following line instead
   // public void add_Event(ValueChanged p)
   {
      ev = (ValueChanged) System.Delegate.Combine(ev, p);
   }

   /** @event */
   public void remove_Event(ValueChanged p)
   {
      ev = (ValueChanged) System.Delegate.Remove(ev, p);
   }
}
```

コンパイラ エラー VJS1811

エラー メッセージ
Event アクセサは、add_ または remove_ で始めなければなりません。

Event アクセサ名の先頭の文字が、予期した文字ではありません。

次の例では、VJS1811 エラーが生成されます。

```
// vjs1811.jsl
// compile with: /target:library
/** @delegate  */
public delegate void ValueChanged();

public class Container
{
   public ValueChanged ev = null;

   /** @event */
   public void set_Event(ValueChanged p)   // VJS1811
   // try the following line instead
   // public void add_Event(ValueChanged p)
   {
      ev = (ValueChanged) System.Delegate.Combine(ev, p);
   }

   /** @event */
   public void remove_Event(ValueChanged p)
   {
      ev = (ValueChanged) System.Delegate.Remove(ev, p);
   }
}
```

コンパイラ エラー VJS1812

エラー メッセージ
型 ubyte、short、int または long が必要です

列挙型で、無効な型が使用されています。列挙型 (Enum) のメンバは、列挙値として使用可能な値を表すため、**char** 以外の整数型である必要があります。

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1817

エラー メッセージ
プロパティ インデックス 'インデックス' は、型 'int' でなければなりません

int 以外のプロパティ インデックスが指定されています。

このエラーを解決するには

1. 取得または設定しようとしているプロパティを見つけます。
2. パラメータの型を **int** に変更します。

使用例

次の例では、VJS1817 エラーが生成されます。

```
package ConsoleApplication;

/**
 * Summary description for Program
 */
public class Program
{
    public static void main(String[] args)
    {
        //
        // Add code to start application here:
        //
    }

    public class WithNonIntIndexer
    {
        protected String[] __prop = new String[3];

        /**@beanproperty*/
        public void setprop(String index, String val)
        {
        }

        /**@beanproperty*/
        public String getprop(String index)
        {
            return null;
        }
    }

}
```

参照

関連項目

プロパティの定義と使用

その他の技術情報

Visual J# コンパイラ

コンパイラ エラー VJS1818

エラー メッセージ
プロパティ アクセサは、is、get、または set で始めなければなりません。

Visual J# は、.NET Framework クラスのプロパティを **get_** アクセサ メソッドと **set_** アクセサ メソッドに変換します。取得可能なプロパティの場合は、**get_** で始まるアクセサになります。設定可能なプロパティの場合は、アクセサは **set_** で始まっている必要があります。

このエラーを解決するには

1. 取得または設定しようとしているプロパティを見つけます。
2. 適切なプレフィックス (`is`、`get_`、または `set_`) を使用してアクセサを変更します。

使用例
次の例では、VJS1818 エラーが生成されます。

```
public class UnsupportedNamingConvention
    {
        protected int __prop = 23;

        /**@beanproperty */
        public void putprop(int i)
        {
        }

        public static void main(String[] args)
        {
        }
    }
```

参照
関連項目
プロパティの定義と使用
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1819

エラー メッセージ
プロパティ 'プロパティ' のアクセサにパラメータが多すぎます

指定されたプロパティ宣言の引数に、必要以上のパラメータが存在します。

このエラーを解決するには

1. 指定されたプロパティを見つけます。
2. プロパティのパラメータを確認します。
3. 不要なパラメータを削除します。

使用例
次の例では、VJS1819 エラーが生成されます。

```
public class IsWithArgs
{
/**@beanproperty*/
public boolean ispropWithNonZeroArgs(int i)
{
return false;
}
public static void main(String[] args)
{
}
}
```

参照
関連項目
プロパティの定義と使用
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1820

エラー メッセージ
is アクセサには、boolean の戻り値の型が必要です

is アクセサは、チェック対象のアイテムがプロパティであるかどうかを指定するものです。戻り値の型は boolean であることが必要です。

このエラーを解決するには

1. **is<property>** メソッドを探します。
2. 戻り値の型を変更します。

使用例
次の例では、VJS1820 エラーが生成されます。

```
public class IsWithNonBooleanReturnType
{

    /**@beanproperty */
    public int isprop()
    {
        return 23;
    }

    public static void main(String[] args)
    {
    }
}
```

参照
関連項目
プロパティの定義と使用
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1850

エラー メッセージ
修飾子 '修飾子' はデリゲートでは使用できません。

コードに無効な修飾子が存在します。

このエラーを解決するには

- 無効な修飾子を探して削除します。

使用例
次の例では、VJS1850 エラーが生成されます。

```
public class Test
{
  public strictfp delegate int MyDelegate();
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1851

エラー メッセージ
Value Type 型を volatile として設定できません

値型のフィールドを volatile としてマークすることはできません。

このエラーを解決するには

-

使用例
次の例では、VJS1851 エラーが生成されます。

```
public class Program
{
public static void main(String[] args)
{
//
// TODO: Add code to start application here
//
}

}

public class Field
{
public static volatile V js_volatilefield_js; // VJS1851 expected
}

public final class V extends System.ValueType
{
public int i;
}
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1852

エラー メッセージ
修飾子 '修飾子' は列挙では使用できません

指定された修飾子は、列挙型 (Enum) では使用できません。

このエラーを解決するには

- 該当する修飾子を削除または変更します。

使用例
次の例では、VJS1852 エラーが生成されます。

```
package ConsoleApplication;

/**
 * Summary description for Program
 */
public class Program
{
   public static void main(String[] args)
   {
   }

}
strictfp enum VJS1852 {
}
```

参照
関連項目
ユーザー定義の列挙型 (J#)
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1853

エラー メッセージ
参照 '参照' には、競合するプラットフォーム設定が含まれます

オペレーティング システム プラットフォーム フラグ設定が、コンパイル中の各アプリケーションと一致していません。

このエラーを解決するには

- アプリケーションに対応するオペレーティング システムを指定します。たとえば、`/platform:x64` を指定した場合、64 ビットのオペレーティング システム用にアプリケーションがコンパイルされます。

使用例

既定では、Visual J# コンパイラは、`/platform:x32` でアプリケーションをビルドするため、次の例では、VJS1853 エラーが生成されます。

```
VJS1853a.cs
// compile with: /w:0 /out:VJS1853a.dll /platform:x64 /t:library
// VJS1853 expected
public class Ref
{

}

// VJS1853b.jsl
// compile with: /t:library  /out:VJS1853b.dll /r:VJS1853a.dll

import System.*;

public class Source
{
    public static void main(String[] args) { }

}
```

参照

その他の技術情報

Visual J# コンパイラ

コンパイラ エラー VJS1854

エラー メッセージ
'属性' 属性に対する引数は、有効な識別子でなければなりません。

このエラーを解決するには

-

使用例
次の例では、VJS1854 エラーが生成されます。

```
class ref
   {
      /** @attribute System.Diagnostics.ConditionalAttribute("~DEBUG#") */
      void method1() { }
   }
```

参照
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS1855

エラー メッセージ
列挙 '列挙' を明示的にインスタンス化できません

Enum 型は、暗黙的に static になります。したがって、**new** を使用して、**enum** を明示的にインスタンス化することはできません。

このエラーを解決するには

- 静的な **enum** を参照するには、同じスコープで静的メンバにアクセス可能でない限り、完全修飾名を使用します。

使用例

次の例では、VJS1855 エラーが生成されます。

```
public enum Color { Red, Green, Blue }

public class myClass
{
  public void Foo()
  {
    Color c = new Color();  // VJS1855 expected
  }
}
```

参照

関連項目

ユーザー定義の列挙型 (J#)

その他の技術情報

Visual J# コンパイラ

コンパイラ エラー VJS1856

エラー メッセージ
'値'の定数値の評価により、循環参照が発生します

enum の各メンバには、関連付けられた定数値があります。**enum** の複数のメンバで、同じ関連付けられた値を共有できます。**enum** の各メンバには、コンパイル時に定数を関連付けることができます。ただし、この関連付けによって循環参照が生じないようにする必要があります。この循環制限を除いて、**enum** のメンバ初期化子は、その構文の位置にかかわらず、他の **enum** のメンバ初期化子を自由に参照できます。

このエラーを解決するには

- 循環参照を引き起こしている定数の値を修正します。

使用例
次の例では VJS1856 エラーが生成されます。

```
enum Color {
    Red(Green),
    BLUE,
    Green(Red)
}
```

参照
関連項目
ユーザー定義の列挙型 (J#)
その他の技術情報
Visual J# コンパイラ

コンパイラ エラー VJS2003

エラー メッセージ
#endif ディレクティブが必要です。

#endif ディレクティブがありません。

次の例では、VJS2003 エラーが生成されます。

```
public class MyClass
{
   public static void main()
   {
      #if DEBUG
      // uncomment the following line to resolve
   // #endif
   }
}
```

コンパイラ エラー VJS2004

エラー メッセージ
不適切なプリプロセッサ ディレクティブです。

予測できないディレクティブが検出されました。

次の例では、VJS2004 エラーが生成されます。

```
public class MyClass
{
   public static void main()
   {
      #endif   // no #if
   }
}
```

コンパイラ エラー VJS2008

エラー メッセージ
IDE 制限を越えています。ファイルに入るのは <数値> 行までです。

現在のリリースでは、ソース コード ファイルが 2,097,151 行を超えた場合にこのエラーが生成されます。

コンパイラ エラー VJS2009

エラー メッセージ
IDE 制限を越えています。1 行に入るのは <数値> 文字までです。

現在のリリースでは、ソース コード ファイルが 2,046 文字を超えた場合にこのエラーが生成されます。

コンパイラ エラー VJS2010

エラー メッセージ
#endregion ディレクティブが必要です。

#region ディレクティブが見つかりましたが、**#endregion** ディレクティブがありません。

次の例では、VJS2010 エラーが生成されます。

```
#region
public class MyClass
{
   public static void main()
   {
   }
}
// #endregion
```

コンパイラ エラー VJS2012

エラー メッセージ
無効なプリプロセッサの式です。

プリプロセッサ式は、ブール型であることが必要です。

次の例では、VJS2012 エラーが生成されます。

```
#if 1   // VJS2012
// try the following line instead
#if true
#endif
```

コンパイラ エラー VJS2013

エラー メッセージ
#line に指定されたファイル名は長すぎます。

ファイル名を **#line** ディレクティブに渡す場合、260 文字を超えるファイル名は使用できません。

コンパイラ エラー VJS2015

エラー メッセージ
ファイル名、単一行コメント、または行の終わりが必要です。

不正な形式のステートメントが検出されました。

次の例では、VJS2015 エラーが生成されます。

```
#line 4 hello
// try the following line instead
#line 4 "myfilename"
```

コンパイラ エラー VJS2101

エラー メッセージ
) が必要です。

閉じかっこがありません。

次の例では、VJS2101 エラーが生成されます。

```
public class MyClass
{
   public static void main(
   // try the following line instead
   // public static void main()
   {
   }
}
```

コンパイラ エラー VJS2102

エラー メッセージ
定数値が必要です。

case ステートメントには、定数を渡す必要があります。

次の例では、VJS2102 エラーが生成されます。

```
public class MyClass
{
   public static void main()
   {
      int i = 0;
      switch (i)
      {
         case :   // VJS2102
         // try the following line instead
         // case 0:
         default:
      }
   }
}
```

コンパイラ エラー VJS2103

エラー メッセージ
} が必要です。

右中かっこ (}) がありません。

次の例では、VJS2103 エラーが生成されます。

```
public class MyClass
{
   public static void main()
   {
   }
// }
```

コンパイラ エラー VJS2104

エラー メッセージ
{ ??????

左中かっこ ({) がありません。

次の例では、VJS2104 エラーが生成されます。

```
public class MyClass
{
   public static void main()
   // {
   }
}
```

コンパイラ エラー VJS2105

エラー メッセージ
; が必要です。

セミコロン (;) がありません。

次の例では、VJS2105 エラーが生成されます。

```
public class MyClass
{
   int i
}
```

コンパイラ エラー VJS2107

エラー メッセージ
構文エラーです。'<トークン>' が必要です。

トークンが必要ですが、見つかりません。

次の例では、VJS2107 エラーが生成されます。

```
public class MyClass
{
   public static void main(String[ args)   // VJS2107
   // try the following line instead
   // public static void main(String[] args)
   {
   }
}
```

コンパイラ エラー VJS2120

エラー メッセージ
ID がありません。

ID が見つかりませんでした。

次の例では、VJS2120 エラーが生成されます。

```
public class MyClass
{
   int ;
}
```

コンパイラ エラー VJS2112

エラー メッセージ
型またはコンストラクタが必要です。

不正な形式の宣言が検出されました。

次の例では、VJS2112 エラーが生成されます。

```
public class MyClass
{
   public ()   // VJS2112
   {
   }
}
```

コンパイラ エラー VJS2113

エラー メッセージ
コンストラクタに戻り値の型は適用できません。

コンストラクタに戻り値の型を含めることはできません。

次の例では、VJS2113 エラーが生成されます。

```
public class MyClass
{
   public MyClass() []   // VJS2113, remove array notation
   {
   }
}
```

コンパイラ エラー VJS2116

エラー メッセージ
配列の初期化子が必要です。

配列宣言の形式が不正です。

次の例では、VJS2116 エラーが生成されます。

```
class Test
{
   public static void main(String[] args)
   {
      int[] x = new int[];   // VJS2116
      // try the following line instead
      int[] x2 = new int[10];
   }
}
```

コンパイラ エラー VJS2117

エラー メッセージ
ここでは配列の初期化子を予期していません。

現在のコンテキストでは配列を初期化できません。

次の例では、VJS2117 エラーが生成されます。

```
public class Class1
{
   public static void main(String[] args)
   {
      Class1 cl = new Class1();
      cl.method(new int[2]{});   // VJS2117
      // try the following line instead
      // cl.method(new int[2]);
   }
}
```

コンパイラ エラー VJS2118

エラー メッセージ
型が必要です。

必要な型が見つかりません。

次の例では、VJS2118 エラーが生成されます。

```
class Test
{
   public static void main(String[] args)
   {
      Test x = new Test();
      if (x instanceof )   // VJS2118
      // try the following line instead
      // if (x instanceof Test)
      {
      }
   }
}
```

コンパイラ エラー VJS2119

エラー メッセージ
マルチキャスト デリゲートには、void の戻り値の型を指定しなければなりません。

戻り値の型が正しくないため、マルチキャスト デリゲート宣言の形式が不正です。

次の例では、VJS2119 エラーが生成されます。

```
class MyClass
{
   public multicast delegate int MyDelegate();   // VJS2119
   // try the following line instead
   public multicast delegate void CorrectDelegate();
}
```

コンパイラ エラー VJS2202

エラー メッセージ
キーワード case または default は switch ブロックのコードの前に記述しなければなりません。

switch ブロックで、予期しないステートメントが検出されました。

次の例では、VJS2202 エラーが生成されます。

```
public class MyClass
{
   public static void main()
   {
      int i = 0;
      switch (i)
      {
         i:   // VJS2202, delete
         case 0:
         default:
      }
   }
}
```

コンパイラ エラー VJS2203

エラー メッセージ
式ステートメントとして使用できるのは、代入式、呼び出し、インクリメント、デクリメント、および新しい式のみです。

ステートメントの形式が不正です。

次の例では、VJS2203 エラーが生成されます。

```
public class MyClass
{
   public static void main()
   {
      i;   // VJS2203
   }
}
```

コンパイラ エラー VJS2204

エラー メッセージ
'modifier' 修飾子が重複しています。

宣言ではアクセス修飾子を重複して使用できません。

次の例では、VJS2204 エラーが生成されます。

```
public class MyClass
{
   public public static void main()   // VJS2204, two public modifiers
   {
   }
}
```

コンパイラ エラー VJS2207

エラー メッセージ
配列型に割り当てるには、配列初期化子式のみ使用できます。

配列型でない変数で、配列の初期化を試みました。

次の例では、VJS2207 エラーが生成されます。

```
public class Class1
{
   public static void main(String[] args)
   {
      int IAmNotAnArray = {1, 2};   // VJS2207
      // try the following line instead
      int[] x = new int[]{1,2};
   }
}
```

コンパイラ エラー VJS2208

エラー メッセージ
クラスまたはインターフェイス メソッドは戻り値を持たなければなりません。

メソッド宣言に戻り値の型が含まれていません。

次の例では、VJS2208 エラーが生成されます。

```
public class MyClass
{
   public static main()   // VJS2208
   // try the following line instead
   // public static void main()
   {
   }
}
```

コンパイラ エラー VJS2210

エラー メッセージ
クラス、インターフェイスまたはデリゲートが必要です。

無効なトークンが検出されました。

次の例では、VJS2210 エラーが生成されます。

```
// vjs2210a.jsl
// compile with: /target:library
int i;   // VJS2210
```

このエラーは、同じファイルに複数の package ステートメントを記述した場合にも発生します。

```
// VJS2210b.jsl
// compile with: /target:library
package MyPackage1;
public class X {}

package MyPackage2;   // VJS2210
public class Y {}
```

コンパイラ エラー VJS2211

エラー メッセージ
内部クラスにインターフェイスを含めることはできません。

内部クラスにインターフェイスが含まれていますが、このように使用することは許可されていません。

次の例では、VJS2211 エラーが生成されます。

```
public class MyClass
{
   class MyInnerClass
   {
      interface MyInterface   // VJS2211
      {
      }
   }
}
```

コンパイラ エラー VJS2213

エラー メッセージ
クラスに無効なトークン '<トークン名>' があるか、またはインターフェイス メンバ宣言です。

無効なトークンが、クラスまたはインターフェイスで検出されました。

次の例では、VJS2213 エラーが生成されます。

```
public class MyClass
{
   default;   // VJS2213
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS2301

エラー メッセージ
定数が 2 行目に続いています。

文字またはリテラル文字列で改行文字が見つかりました。

次の例では、VJS2301 エラーが生成されます。

```
class MyClass
{
   char s = '
      f';
   // try the following line instead
   // char s = 'f';
}
```

コンパイラ エラー VJS2302

エラー メッセージ
整数定数が int の範囲外にあります。

整数定数を **int** 型として表すことができません。このエラーは、**int** 定数でオーバーフローが発生した場合に表示されます。

次の例では、VJS2302 エラーが生成されます。

```
class MyClass
{
       int i = 100000000000; // VJS2302
}
```

コンパイラ エラー VJS2303

エラー メッセージ
定数が浮動小数点の値の範囲外にあります。

定数を **float** 型として表すことができません。このエラーは、浮動小数点の定数でオーバーフローやアンダーフローが発生した場合に表示されます。

次の例では、VJS2303 エラーが生成されます。

```
class MyClass
{
   float f = 1111111111111111111111111111111111111111e+2f;   // VJS2303
   // try the following line instead
   // float f = 3.140e+30f;

   public static void main()
   {
   }
}
```

コンパイラ エラー VJS2304

エラー メッセージ
無効な数字です。

無効な形式の数字が検出されました。

次の例では、VJS2304 エラーが生成されます。

```
public class Class1
{
   public static void main(String[] args)
   {
      int a = 0x;   // VJS2304
      // try the following line instead
      // int a = 0xa;
   }
}
```

コンパイラ エラー VJS2305

エラー メッセージ
空の文字リテラルです。

char 型の変数が空の文字に代入されましたが、このように代入することは許可されていません。

次の例では、VJS2305 エラーが生成されます。

```
class MyClass
{
   public static void main()
   {
      char i = '';   // VJS2305
      // try the following line instead
      char j = 'a';
   }
}
```

コンパイラ エラー VJS2306

エラー メッセージ
文字リテラルに文字が多すぎます。

char 型の変数が 2 つ以上の文字に代入されましたが、このように代入することは許可されていません。

VJS2306 エラーは、データ バインディングの実行時にも発生する場合があります。たとえば、次のコード行はエラーになります。

```
<%# DataBinder.Eval(Container.DataItem, 'doctitle') %>
```

代わりに、次のコード行を試してください。

```
<%# DataBinder.Eval(Container.DataItem, "doctitle") %>
```

次の例では、VJS2306 エラーが生成されます。

```
class MyClass
{
   public static void main()
   {
      char i = 'ab';   // VJS2306
      // try the following line instead
      char j = 'a';
   }
}
```

コンパイラ エラー VJS2307

エラー メッセージ
認識できないエスケープ シーケンスです。

文字の形式が不正です。

次の例では、VJS2307 エラーが生成されます。

```
class MyClass
{
   char s = '\8';   // VJS2307, not a valid octal char
   char c = '\7';   // OK
}
```

コンパイラ エラー VJS2308

エラー メッセージ
定数が double 値の範囲外にあります。

定数を **double** 型として表すことができません。このエラーは、**double** 型の定数でオーバーフローやアンダーフローが発生した場合に表示されます。

次の例では、VJS2308 エラーが生成されます。

```
public class Class1
{
   public static void main(String[] args)
   {
      double d =  4e3045d;   // VJS2308, try a smaller value
   }
}
```

コンパイラ エラー VJS2309

エラー メッセージ
整数定数が long の範囲外にあります。

整数定数を **long** 型として表すことができません。このエラーは、**long** でオーバーフローが発生した場合に表示されます。

次の例では、VJS2309 エラーが生成されます。

```
class MyClass
{
   long s = 10000000000000000000000000L; // VJS2309
}
```

コンパイラ エラー VJS2401

エラー メッセージ
インターフェイスを final に指定することはできません。

インターフェイス定義に **final** アクセス修飾子が使用されていますが、このように使用することは許可されていません。

次の例では、VJS2401 エラーが生成されます。

```
final interface MyInterface   // VJS2401
{
}
```

コンパイラ エラー VJS2402

エラー メッセージ
修飾子 '<修飾子名>' をクラスに対して使用することはできません。

クラス宣言で使用できない修飾子が、クラス宣言で使用されています。

次の例では、VJS2402 エラーが生成されます。

```
volatile class MyClass   // VJS2402
{
}
```

コンパイラ エラー VJS2403

エラー メッセージ
修飾子 '<修飾子名>' をインターフェイスに対して使用することはできません。

修飾子が、インターフェイス宣言で正しく使用されていません。

次の例では、VJS2403 エラーが生成されます。

```
volatile interface inter1   // VJS2403
{
}
```

コンパイラ エラー VJS2404

エラー メッセージ
修飾子 '<修飾子名>' をデリゲートに対して使用することはできません。

修飾子が、デリゲート宣言に正しく適用されていません。

次の例では、VJS2404 エラーが生成されます。

```
volatile delegate void Del();   // VJS2404
```

コンパイラ エラー VJS2405

エラー メッセージ
静的クラスは内部クラスでは定義できません。

外側のクラスがスタティックでない場合は、スタティック内部クラスを定義できません。

次の例では、VJS2405 エラーが生成されます。

```
class Class1
{
   class InnerClass   // not static
   {
      static class InnnerMostClass   // VJS2405
      {
      }
   }
}
```

コンパイラ エラー VJS2406

エラー メッセージ
修飾子 '<修飾子名>' をローカル クラスに対して使用することはできません。

修飾子が、ローカル クラス宣言または内部クラス宣言で正しく使用されていません。

次の例では、VJS2406 エラーが生成されます。

```
class Class1
{
   public static void main(String[] args)
   {
      abstract static class LocalClass   // VJS2406, static not allowed here
      {
      }
   }
}
```

コンパイラ エラー VJS2407

エラー メッセージ
修飾子 '<修飾子名>' をコンストラクタに対して使用することはできません。

修飾子 (<修飾子名>) が、コンストラクタに正しく適用されていません。

次の例では、VJS2407 エラーが生成されます。

```
class Class1
{
   static public Class1()   // VJS2407, static not allowed here
   {
   }
}
```

コンパイラ エラー VJS2408

エラー メッセージ
修飾子 '<修飾子名>' をメソッドに対して使用することはできません。

修飾子 (<修飾子名>) が、メソッド宣言に正しく適用されていません。

次の例では、VJS2408 エラーが生成されます。

```
class Class1
{
   public volatile static void main(String[] args)   // VJS2408, volatile not allowed here
   {
   }
}
```

コンパイラ エラー VJS2409

エラー メッセージ
修飾子 '<修飾子名>' はフィールドには使用できません。

修飾子 (<修飾子名>) が、フィールド宣言に正しく適用されていません。

次の例では、VJS2409 エラーが生成されます。

```
class Class1
{
   synchronized int FieldCantBeSync;   // VJS2409
}
```

コンパイラ エラー VJS2410

エラー メッセージ
'<修飾子名 1>' および '<修飾子名 2>' 修飾子の両方を指定することはできません。

修飾子 (<修飾子名 1>) が、他の修飾子と競合しています。競合している修飾子を削除します。

次の例では、VJS2410 エラーが生成されます。

```
// vjs2410.jsl
// compile with: /target:library
public abstract final class Class1   // VJS2410
{
}
```

コンパイラ エラー VJS2411

エラー メッセージ
修飾子 '<修飾子名>' をインターフェイスのメソッドに対して使用することはできません。

修飾子 (<修飾子名>) が、インターフェイスのメソッド宣言に正しく適用されていません。

次の例では、VJS2411 エラーが生成されます。

```
interface inter1
{
   final void method();   // VJS2411
}
```

コンパイラ エラー VJS2413

エラー メッセージ
修飾子 '<修飾子名>' をインターフェイスのフィールドに対して使用することはできません。

修飾子 (<修飾子名>) が、インターフェイスのフィールド宣言に正しく適用されていません。

次の例では、VJS2413 エラーが生成されます。

```
interface inter1
{
   volatile int a;   // VJS2413
}
```

コンパイラ エラー VJS2414

エラー メッセージ
'<修飾子名 1>' および '<修飾子名 2>' 修飾子の両方を指定することはできません。

同時に指定できない 2 つの修飾子が、同じ宣言に適用されています。

次の例では、VJS2414 エラーが生成されます。

```
class Class1
{
   final volatile int a;   // VJS2414
}
```

コンパイラ エラー VJS2416

エラー メッセージ
外部クラスに '<修飾子名>' 修飾子を指定することはできません。

修飾子 <修飾子名> が、外部クラスの宣言に正しく適用されていません。

次の例では、VJS2416 エラーが生成されます。

```
protected class MyClass   // VJS2416
{
   public static void main()
   {
   }
}
```

コンパイラ エラー VJS2417

エラー メッセージ
外部クラスに '修飾子' 修飾子を指定することはできません。

修飾子 <修飾子名> が、外部インターフェイスの宣言に正しく適用されていません。

次の例では、VJS2417 エラーが生成されます。

```
private interface inter   // VJS2417
{
}
```

コンパイラ エラー VJS2501

エラー メッセージ
型 '型' には、ジェネリック型またはメソッド 'メソッド' 内でパラメータ 'パラメータ' として使用するために、パブリックのパラメータを持たないコンストラクタを指定しなければなりません。

ジェネリック型またはジェネリック メソッドのパラメータとして指定された型は、対応するコンストラクタがパブリックとして宣言されていないか、コンストラクタにパラメータが存在します。

このエラーを解決するには

1. 型パラメータが指定されているメソッドまたはジェネリック型を探します。
2. パラメータをパブリック コンストラクタの型に変更します。または、この型をパラメータとして使用する場合は、コンストラクタにパラメータが存在しないことを確認します。

使用例
次の例では、VJS2501 エラーが生成されます。

```
// VJS2501.cs
// compile with: /target:library
using System;
public class MyClass{}
public class Gen<G> where G:MyClass,new()
{
   public G method<U>(int a)
   {
      G t=default(G);
      return t;
   }
}

public class MyClassExt
{
   public void meth1<T, U, V> (T t, U u, V v) {}
   public void meth1<T, U> ( T t, U u, int i ) {}
   public void meth1<T, U> ( T t, U u,long i ) {}
}

public class Gen2<T,U>
{
   public static int val = 0;
   public Gen2()
   {
      val ++;
   }
}

public class Gen3<T> where T:class {}
public class Gen4<T> where T:struct {}
```

```
// VJS2501_b.jsl
// compile with: /reference:VJS2501.dll
public class MyClassext extends MyClass
{
   private MyClassext() {}
}

public class jsTest
{
   public static void main()
   {
      Gen<MyClassext> mygen = new Gen<MyClassext>();   // VJS2501 expected
```

```
    }
}
```

参照

その他の技術情報

Visual J# コンパイラ
ジェネリックのサンプル (ジェネリックの使用)

コンパイラ エラー VJS2502

エラー メッセージ
型 'type1' は、ジェネリック型またはメソッド 'メソッド' 内でパラメータ 'パラメータ' として使用するために、'type2' に変換可能でなければなりません。

パラメータとして使用している型は、指定されたメソッドのパラメータとして必要な型には変換できません。

このエラーを解決するには

- 指定されたメソッドの型に変換可能な型をパラメータとして使用していることを確認してください。

使用例
次の例では、VJS2502 エラーが生成されます。

```
// VJS2502.cs
// compile with: /target:library
using System;
public class MyClass{}
public class Gen<G> where G:MyClass,new()
{
   public G  method<U>(int a)
   {
      G t=default(G);
      return t;
   }
}
public class MyClassExt
{
   public void meth1<T, U, V> (T t, U u, V v){}
   public void meth1<T, U> ( T t, U u, int i ){}
    public void meth1<T, U> ( T t, U u,long i ){}
}
public class Gen2<T,U>
{
   public static int val = 0;
   public Gen2()
   {
      val ++;
   }
}
public class Gen3<T>  where T:class {}
public class Gen4<T>  where T:struct{}
```

```
// VJS2502.jsl
// compile with: /reference:VJS2502.dll
public class jsTest
{
   public static void main()
   {
      Gen<int> mygen = new Gen<int>();   // VJS2502
   }
}
```

参照
その他の技術情報
Visual J# コンパイラ
ジェネリックのサンプル (ジェネリックの使用)

コンパイラ エラー VJS2504

エラー メッセージ
キーワード 'キーワード' はこのコンテキストで使用できません

指定のキーワードを不正に使用しました。

使用例

```
// VJS2504.cs
// compile with: /target:library
public class MyClass{}
public class Gen<G> where G:MyClass,new()
{
   public G  method<U>(int a)
   {
      G t=default(G);
      return t;
   }
}

public class MyClassExt
{
   public void meth1<T, U, V> (T t, U u, V v) {}
   public void meth1<T, U> ( T t, U u, int i ) {}
   public void meth1<T, U> ( T t, U u,long i ) {}
}

public class Gen2<T,U>
{
   public static int val = 0;
   public Gen2()
   {
      val ++;
   }
}

public class Gen3<T> where T:class {}
public class Gen4<T> where T:struct {}
```

次の例では、VJS2504 エラーが生成されます。

```
// VJS2504.jsl
// compile with: /reference:VJS2504.dll
public class MyClassext extends MyClass
{
   private MyClassext(){}
}

public class jsTest
{
   public static void main()
   {
      Gen<void> mygen = new Gen<void>();   // VJS2504 expected
   }
}
```

参照

その他の技術情報

Visual J# コンパイラ
ジェネリックのサンプル (ジェネリックの使用)

コンパイラ エラー VJS2505

エラー メッセージ
型 '型' は 型引数として使用できません

型引数としては使用できない型が指定されています。

このエラーを解決するには

- 代わりにネイティブ型を使用します。たとえば、**int** などです。

使用例

```
// VJS2505.cs
// compile with: /target:library
public class MyClass {}

public class Gen<G> where G : MyClass, new()
{
   public G method<U>(int a)
   {
      G t = default(G);
      return t;
   }
}

public class MyClassExt
{
   public void meth1<T, U, V>(T t, U u, V v) {}
   public void meth1<T, U>(T t, U u, int i) {}
   public void meth1<T, U>(T t, U u, long i) {}
}

public class Gen2<T, U>
{
   public static int val = 0;
   public Gen2()
   {
      val++;
   }
}

public class Gen3<T> where T : class {}
public class Gen4<T> where T : struct {}
```

次の例では、VJS2505 エラーが生成されます。

```
// VJS2505_b.jsl
// compile with: /reference:VJS2505.dll
import System.*;
public class MyClassext extends MyClass
{
   private MyClassext() {}
}

public class jsTest
{
   public static void main()
   {
      Gen<RuntimeArgumentHandle> mygen = new Gen<RuntimeArgumentHandle>();   // VJS2505
   }
}
```

参照

その他の技術情報

Visual J# コンパイラ

ジェネリックのサンプル (ジェネリックの使用)

コンパイラ エラー VJS2506

エラー メッセージ
.NET 言語拡張が無効になっているときは、ジェネリックを使用できません

ジェネリックを使用するには、.NET 言語拡張機能が必要です。

このエラーを解決するには

- ジェネリックを使用してプロジェクトをコンパイルするには、あらかじめ .NET 拡張機能を有効にしておく必要があります。

使用例
次の例では、VJS2506 エラーが生成されます。

```
// VJS2506.jsl
// compile with: /t:l /x:net
package ConsoleApplication;
//import System.Collections.Generic.*;
public class Program
{
   public static void main(String[] args)
}
public class VJS2506
{
   public Stack<int> mystack = new Stack<int>();
// VJS2506 expected
}
```

参照
その他の技術情報
Visual J# コンパイラ
ジェネリックのサンプル (ジェネリックの使用)

コンパイラ エラー VJS2508

エラー メッセージ
アクセシビリティに一貫性がありません。型 '型'、またはその型引数にアクセスできません。

指定された型にアクセスできません。

このエラーを解決するには

- アクセスしようとしている型が private または restricted 以外であることをなんらかの方法で確認します。

使用例
次の例では VJS2508 エラーが生成されます。

```
// VJS2508.cs
// compile with: /target:library
using System;
public class MyClass{}
public class Gen<G> where G:MyClass,new()
{
   public G  method<U>(int a)
   {
      G t=default(G);
      return t;
   }
}
public class MyClassExt
{
   public void meth1<T, U, V> (T t, U u, V v){}
   public void meth1<T, U> ( T t, U u, int i ){}
    public void meth1<T, U> ( T t, U u,long i ){}
}
public class Gen2<T,U>
{
   public static int val = 0;
   public Gen2()
   {
      val ++;
   }
}
public class Gen3<T>  where T:class {}
public class Gen4<T>  where T:struct{}
```

```
// VJS2508_b.jsl
// compile with: /reference:VJS2508.dll
import System.*;
import java.util.Date;
import System.Collections.Generic.*;

public class jScope
{
      public Stack<int> mystack;
      public class CPub1 {}
      public class CPub2 {}
      private  class CPri1{}
      protected class CProt1{}
      //public types
      public  Gen2<CPub1,CPub2> pubfld1;
      public  Gen2<CPub1,CPub2> pubfld2;
      //One type protected
      public Gen2<CProt1, CPub1> protfld1;
      //Private type
      public Gen2<CPri1,CPub1> prifld1;
```

```
        //Private type
        public Gen2<CPri1,CPub1> getPri()
        {
              return new Gen2<CPri1,CPub1>();
        }
        //One type protected
        public  Gen2<CProt1, CPub1> getProt()
        {
              return new  Gen2<CProt1, CPub1>();
        }
        //public types
        public Gen2<CPub1,CPub2> getPub()
        {
              return new Gen2<CPub1,CPub2> ();
  // VJS2508 expected
        }
  }
```

参照

その他の技術情報

Visual J# コンパイラ

ジェネリックのサンプル (ジェネリックの使用)

コンパイラ エラー VJS2509

エラー メッセージ
型 '型' は、ジェネリック型のパラメータ 'パラメータ'、またはメソッド 'メソッド' として使用するために、参照型でなければなりません。

ジェネリック型またはジェネリック メソッドで指定されたパラメータが参照型ではありません。

このエラーを解決するには

- 指定した型をチェックし、参照型に変更できるかどうかを確認します。

使用例

```
// VJS2509.cs
// compile with: /target:library
public class MyClass {}
public class Gen<G> where G:MyClass,new()
{
   public G  method<U>(int a)
   {
      G t=default(G);
      return t;
   }
}
public class MyClassExt
{
   public void meth1<T, U, V> (T t, U u, V v) {}
   public void meth1<T, U> ( T t, U u, int i ) {}
   public void meth1<T, U> ( T t, U u,long i ) {}
}

public class Gen2<T,U>
{
   public static int val = 0;
   public Gen2()
   {
      val ++;
   }
}

public class Gen3<T> where T:class {}
public class Gen4<T> where T:struct {}
```

```
// VJS2509_b.jsl
// compile with: /reference:VJS2509.dll
public class jsTest
{
   public static void main()
   {
      Gen3<int> mygen = new Gen3<int>();   // VSJ2509
   }
}
```

参照

その他の技術情報
Visual J# コンパイラ
ジェネリックのサンプル (ジェネリックの使用)

コンパイラ エラー VJS2510

エラー メッセージ
型 '型' は、ジェネリック型のパラメータ 'パラメータ'、またはメソッド 'メソッド' として使用するために、値型でなければなりません

指定されたメソッドのパラメータは、値型であることが必要です。

使用例
次の例では、VJS2510 エラーが生成されます。

```
// VJS2510.cs
// compile with: /target:library
using System;
public class MyClass {}

public class Gen<G> where G:MyClass,new()
{
   public G method<U>(int a)
   {
      G t=default(G);
      return t;
   }
}

public class MyClassExt
{
   public void meth1<T, U, V> (T t, U u, V v) {}
   public void meth1<T, U> ( T t, U u, int i ) {}
   public void meth1<T, U> ( T t, U u,long i ) {}
}

public class Gen2<T,U>
{
   public static int val = 0;
   public Gen2()
   {
      val ++;
   }
}

public class Gen3<T> where T:class {}
public class Gen4<T> where T:struct {}
```

```
// VJS2510_b.jsl
// compile with: /reference:VJS2510.dll
import java.util.*;
public class jsTest
{
   public static void main()
   {
      Gen4<int> mygen = new Gen4<int>();
      Gen4<Stack> nonVal = new Gen4<Stack>();   // VSJ2510 expected
   }
}
```

参照
その他の技術情報
Visual J# コンパイラ
ジェネリックのサンプル (ジェネリックの使用)

コンパイラ エラー VJS2701

エラー メッセージ
アセンブリ 'アセンブリ, Version=バージョン, Culture=カルチャ, PublicKeyToken=publickeytoken' が見つかりません。

指定されたアセンブリが見つかりません。このアセンブリは、メタデータ内のいずれかのファイルによって参照されています。一般に、この問題は、ビルド エラーが発生したか、アセンブリが正しい順序でビルドされなかった場合に起こります。

このエラーを解決するには

1. 各アセンブリが、正しい DLL 名または EXE 名でビルドされるようにします。
2. メイクファイルまたは Visual Studio で、ビルドの順番と依存関係が正しいかどうかを確認します。

使用例

このサンプルをコンパイルする前に、VJS2701_key.snk というキーファイルを作成してください。詳細については、「/keyfile (暗号化キーのファイル名の指定)」を参照してください。

次の例では、VJS2701 エラーが生成されます。

```
// vjs2701_1.jsl
// compile with: /target:library /keyfile:VJS2701_key.snk /out:VJS2701a.dll
/** @assembly System.Reflection.AssemblyVersion("1.0.0.0")*/
/** @assembly System.Reflection.AssemblyCulture("de")*/
public class B {
   public int myAdd(int i, int j)
   {
      return  i + j;
   }
}
```

```
// vjs2701_2.jsl
// compile with: /target:library /reference:VJS2701a.dll /keyfile:VJS2701_key.snk /out:VJS2
701b.dll
public class D {
   public static B getB()
   {
      return new B();
   }
}
```

```
// vjs2701_3.jsl
// compile with: /target:library /keyfile:VJS2701_key.snk /out:VJS2701a.dll

// Notice that the compile comment builds to the wrong DLL name.
// Should build to VJS2701a2.dll.
/** @assembly System.Reflection.AssemblyVersion("1.0.0.0")*/
/** @assembly System.Reflection.AssemblyCulture("en-us")*/
public class B {
   public int myAdd(int i, int j)
   {
      return  i + j;
   }
}
```

```
// vjs2701_4.jsl
// compile with: /target:library /reference:VJS2701a.dll /reference:VJS2701b.dll /keyfile:V
JS2701_key.snk /out:VJS2701c.dll
// VJS2701 expected
public class Test {
   public static void main() {
```

```
        B b2;
        D d = new D();
        b2 = d.getB();
    }
}
```

参照

その他の技術情報

Visual J# コンパイラ

コンパイラ エラー VJS2702

エラー メッセージ
検出されたアセンブリ assembly1 は、必要なアセンブリ assembly2 よりも古いバージョンです。

assembly1 と *assembly2* とでアセンブリのバージョン番号が一致していません。

このエラーを解決するには

1. 関連するすべてのアセンブリが同じバージョン番号となるように **AssemblyVersion** 属性を修正します。たとえば、`/**@assembly System.Reflection.AssemblyVersion("2.2.2.2")*/` となります。
2. すべてのアセンブリを再コンパイルします。

使用例
このコード例をコンパイルする前に、VJS2702_key.snk というキーファイルを生成します。

詳細については、「/keyfile (暗号化キーのファイル名の指定)」を参照してください。

次の例では、VJS2702 エラーが生成されます。

```
// vjs2702a2.jsl
// compile with: /t:library /keyfile:VJS2702_key.snk /out:VJS2702a.dll
/**@assembly System.Reflection.AssemblyVersion("2.2.2.2")*/
/**@assembly System.Reflection.AssemblyCultureAttribute("de")*/
public class B {
    //myAdd() of v1 returns 'i + j + 1' and myAdd of V2 returns 'i + j + 2'
    public int myAdd(int i, int j) {
        return i + j + 2;
    }
}
```

```
// vjs2702b.jsl
// compile with: /t:library /r:VJS2702a.dll /keyfile:VJS2702_key.snk /out:VJS2702b.dll
public class D {
    public B getB() {
        return new B();
    }
}
```

```
// vjs2702a1.jsl
// compile with: /t:library /keyfile:VJS2702_key.snk /out:VJS2702a.dll
/**@assembly System.Reflection.AssemblyVersion("2.1.2.2")*/
/**@assembly System.Reflection.AssemblyCultureAttribute("de")*/
public class B {
    //myAdd() of v1 returns 'i + j + 1' and myAdd of V2 returns 'i + j + 2'
    public int myAdd(int i, int j) {
        return i + j + 1;
    }
}
```

```
// vjs2702c.jsl
// compile with: /t:library /r:VJS2702a.dll /r:VJS2702b.dll /keyfile:VJS2702_key.snk /out:V
JS2702c.dll
public class Test {
    public static void main() {
        B b2;
        D d = new D();
        b2 = d.getB();
        //int ret = b2.myAdd(12, 14);
```

```
        }
    }
```

参照

その他の技術情報

Visual J# コンパイラ

Visual J# Class Library

The J# Class Libraries is Microsoft's implementation of the JDK 1.1.4 specification. Most classes implement functionality that is subset or equivalent to JDK1.1.4 whereas some implement functionality that is subset or equivalent to 1.2. Visual J# 2005 also supports many of the Microsoft Extensions defined in Visual J++ 6.0, such as com.ms.lang.Delegate. For a complete list of which features are supported, see Supported Class Libraries. For a list of the features that Visual J# does not support, see Unsupported Class Libraries and Features.

The J# Class Libraries are built on top of the .NET Framework. The root class of the J# Class Libraries, java.lang.Object, is derived from Object. This means that all J# objects can be used where a .NET object is expected, and vice versa. For example, a java.lang.String can be inserted into a ArrayList. Conversely, a String can be inserted into a java.util.Vector. For more information on the relationship between Visual J# and the .NET Framework, see Visual J# Architecture.

For Visual J# 2005, most packages defined by the Java 1.1.4 specification are supported, but only the most important classes in the following packages are documented at this time:

- java.io (J#)
- java.lang
- java.lang.Class
- java.lang.Object
- java.lang.reflect
- java.lang.String
- java.lang.Thread
- java.lang.ThreadGroup
- java.security.acl
- java.sql
- java.text
- java.util

See Also

Other Resources

Visual J# Reference

java.io (J#)

Contains classes used for data input and output.

Classes

Class	Description
BufferedInputStream	Used to hold data from an input stream. The member methods read data from the buffer.
BufferedOutputStream	Stores the data to be written into a buffer. The data is actually written when the buffer is full or when you empty the buffer by calling the flush method.
BufferedReader	Provides an implementation of the abstract Reader class that buffers character data from another reader.
BufferedWriter	Provides an implementation of the abstract Writer class that buffers character data to another writer.
ByteArrayInputStream	Contains an internal buffer that stores the bytes read from the input stream. By using a counter, it keeps track of the next byte to be read.
ByteArrayOutputStream	Extends the OutputStream class in which the data is written into a byte array. It contains a buffer that automatically grows as data is written to it. It also contains methods to retrieve and convert data.
CharArrayReader	Provides an implementation of the abstract Reader class specific to reading arrays of characters.
CharArrayWriter	Provides an implementation of the abstract Writer class specific to writing into an array of characters. The array can be used to retrieve the written character data as a character array or a string or to write the character buffer to another instance of a Writer object.
EOFException	The exception that is thrown when an attempt is made to read past the end of a file.
File	Contains methods to access, create, or edit files and folders on disk.
FileNotFoundException	The exception that is thrown when an attempt is made to access a file that does not exist.
FileReader	Provides an implementation of the abstract Reader class used to provide a character stream for files.
FileWriter	Provides an implementation of the abstract Writer class specific to writing files.
InputStreamReader	Provides an implementation of the abstract Reader class specific to reading characters in an InputStream object.
InterruptedIOException	The exception that is thrown when a timeout has occurred or there was an interruption during an I/O operation.
InvalidClassException	The exception that is thrown when an attempt is made to serialize or deserialize an instance of a class that does not support serialization.
InvalidObjectException	The exception that is thrown when an error occurs during deserialization of an object.
IOException	The exception that is thrown when a generic I/O error has occurred. This class represents the base class for most of the exceptions in the java.io package.

LineNumberReader	Provides an implementation of the abstract Reader class that counts the number of lines in a stream of data.
OutputStreamWriter	Provides an implementation of the abstract Writer class specific to writing characters to an OutputStream object.
PipedReader	Provides an implementation of the abstract Reader class specific to reading characters from a PipedWriter object.
PipedWriter	Provides an implementation of the abstract Writer class specific to writing characters to a PipedWriter object.
PrintWriter	Provides an implementation of the abstract Writer class specific to printing various primitive types and other objects to an underlying output stream.
PushbackReader	Provides an implementation of the abstract Reader class that buffers data from another stream. The internal buffer is initially filled with the contents read in from the underlying stream. Characters in this internal buffer can then be "pushed back", or replaced with other characters at the position marked.
RandomAccessFile	Represents a file used to store data that can be accessed in a non-sequential manner.
Reader	An abstract class representing the base for all character readers in the java.io package.
StringReader	Provides an implementation of the abstract Reader class used to read characters from a given string.
StringWriter	Provides an implementation of the abstract Writer class used to write characters into a StringBuffer object that can be resized as required.
Writer	An abstract class representing the base for all character writers in the java.io package.

BufferedInputStream Class

Used to hold data from an input stream. The member methods read data from the buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.BufferedInputStream
    extends java.io.FilterInputStream
```

Inheritance Hierarchy

java.lang.Object

java.io.InputStream

java.io.FilterInputStream

java.io.BufferedInputStream

See Also

Concepts

BufferedInputStream Members

java.io

BufferedInputStream Members

Used to hold data from an input stream. The member methods read data from the buffer.

The following tables list the members exposed by the BufferedInputStream type.

Public Constructors

Name	Description
BufferedInputStream	Overloaded.

Public Fields

Name	Description
buf	Stores the array of bytes read from the input stream.
count	Stores the number of bytes that have been read.
in	(inherited from FilterInputStream)
marklimit	Stores the maximum number of bytes that can be marked.
markpos	Stores the position of the marked byte.
pos	Stores the position of the last read byte.

Public Methods

Name	Description
available	Overridden. Returns the available number of bytes that can be read from the input stream without being bloc ked.
close	Overridden. Closes the input buffer.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Marks the current position of the byte read so far in the input buffer.
markSupported	Overridden. Checks if fc2bb7b1-e8ab-44ca-a9a1-362d98852478 and bde7d3cb-ef57-4231-9a8d-4578236481e1 are supported.
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden. Repositions the mark created by fc2bb7b1-e8ab-44ca-a9a1-362d98852478 to its last position w hen mark() was last called.
skip	Overridden. Skips reading a specific number of bytes.

toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BufferedInputStream Class

Concepts

java.io Package

BufferedInputStream Fields

Public Fields

Name	Description
buf	Stores the array of bytes read from the input stream.
count	Stores the number of bytes that have been read.
in	(inherited from FilterInputStream)
marklimit	Stores the maximum number of bytes that can be marked.
markpos	Stores the position of the marked byte.
pos	Stores the position of the last read byte.

See Also

Reference

BufferedInputStream Class

Concepts

java.io Package

BufferedInputStream.buf Field

Stores the array of bytes read from the input stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected byte[] buf;
```

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.count Field

Stores the number of bytes that have been read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int count;
```

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.marklimit Field

Stores the maximum number of bytes that can be marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int marklimit;
```

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.markpos Field

Stores the position of the marked byte.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int markpos;
```

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.pos Field

Stores the position of the last read byte.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int pos;
```

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream Constructor

Overload List

Name	Description
BufferedInputStream (InputStream)	Constructs a repository to read data from a specific stream using the default buffer size (2K).
BufferedInputStream (InputStream, int)	Constructs a repository to read data from a specific stream using a specific buffer size.

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream Constructor (InputStream)

Constructs a repository to read data from a specific stream using the default buffer size (2K).

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.BufferedInputStream(
    java.io.InputStream in);
```

Parameters

in

The input stream.

Example

```
// Read the file name as a program argument:
File f = new File(args[0]);

// Construct an input stream using the file "f":
FileInputStream fin = new FileInputStream(f);

// Construct the buffer using the stream "fin":
BufferedInputStream bin = new BufferedInputStream(fin);
```

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream Constructor (InputStream, Int32)

Constructs a repository to read data from a specific stream using a specific buffer size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.BufferedInputStream(
    java.io.InputStream in,
    int size);
```

Parameters

in

The specified input stream.

size

The size of the buffer.

See Also
Reference
BufferedInputStream Class
Concepts
BufferedInputStream Members
java.io Package

BufferedInputStream Methods

Public Methods

Name	Description
available	Overridden. Returns the available number of bytes that can be read from the input stream without being blocked.
close	Overridden. Closes the input buffer.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Marks the current position of the byte read so far in the input buffer.
markSupported	Overridden. Checks if fc2bb7b1-e8ab-44ca-a9a1-362d98852478 and bde7d3cb-ef57-4231-9a8d-4578236481e1 are supported.
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden. Repositions the mark created by fc2bb7b1-e8ab-44ca-a9a1-362d98852478 to its last position when mark() was last called.
skip	Overridden. Skips reading a specific number of bytes.
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BufferedInputStream Class

Concepts

java.io Package

BufferedInputStream.available Method

Returns the available number of bytes that can be read from the input stream without being blocked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int available() throws java.io.IOException;
```

Return Value

The number of bytes that can be read from the input stream without being blocked.

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.close Method

Closes the input buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void close() throws java.io.IOException;
```

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.mark Method

Marks the current position of the byte read so far in the input buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void mark(
    int readLimit);
```

Parameters

readLimit

The number of bytes to be read before changing the mark position.

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.markSupported Method

Checks if fc2bb7b1-e8ab-44ca-a9a1-362d98852478 and bde7d3cb-ef57-4231-9a8d-4578236481e1 are supported.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

Return Value

true if mark() and reset() are supported; false otherwise.

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.read Method

Overload List

Name	Description
BufferedInputStream.read ()	Reads the next byte of data from the buffered input stream. If the end of the stream has been reached, it returns -1.
BufferedInputStream.read (byte[])	
BufferedInputStream.read (byte[], int, int)	Reads up to the number of bytes of data stipulated by len from the buffered input stream into an array of bytes.

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.read Method ()

Reads the next byte of data from the buffered input stream. If the end of the stream has been reached, it returns -1.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int read() throws java.io.IOException;
```

Return Value

The actual number of bytes that have been read.

Example

The following example reads the file specified as an argument. For example, if the code example is called "readfile," you can read a file named "my-filename.txt," by invoking the program like this:

> readfile my-filename.txt.

The read file will be displayed on the screen.

```
// BIS-Readfile1.jsl
// BufferedInputStream.read example

import java.io.*;

public class myClass
{
    public static void main(String [] args)
    {
        try
        {
            // Accept the name of the input file from the keyboard
            // as an argument:
            File f = new File(args[0]);

            // Declare and initialize the input stream:
            FileInputStream fin = new FileInputStream(f);

            // Declare and initialize the input buffer:
            BufferedInputStream bin = new BufferedInputStream(fin);

            // Read from the buffer one character at a time:
            int c = 0;
            while ((c = bin.read()) >= 0)
            {
                // Display the read character:
                System.out.print((char) c);
            }

            // Close the buffer:
            bin.close();
        }
        catch (IOException e)
        {
            // Do nothing.
        }
    }
}
```

Remarks

This method is blocked until input data is available, the end of the stream is detected, or an exception is thrown.

This method can throw java.io.IOException.

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.read Method (SByte[], Int32, Int32)

Reads up to the number of bytes of data stipulated by len from the buffered input stream into an array of bytes.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int read(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

Buffer into which the data is read.

off

The offset that indicates the starting point of the data.

len

Number of bytes to read.

Return Value

The actual number of bytes read.

Example

The following example reads the file specified as an argument. For example, if the code example is called "readfile," you can read a file named "my-filename.txt," by invoking the program like this:

> readfile my-filename.txt.

```
// BIS-Readfile2.jsl
// Reading from a buffered stream

import java.io.*;

public class myClass
{
    public static void main(String [] args)
    {
        try
        {
            // Accept the name of the input file from the keyboard
            // as a program argument:
            File f = new File(args[0]);

            // Declare and initialize the input stream:
            FileInputStream fin = new FileInputStream(f);

            // Declare and initialize the input buffer:
            BufferedInputStream bin = new BufferedInputStream(fin);

            int c = 0;

            // Declare the buffer and initialize its size:
            byte[] b = new byte[1024];

            // Read from the buffer one character at a time:
            while ((c = bin.read(b, 0, b.length)) >= 0)
            {
```

```
                for (int i = 0; i < c; i++)
                {
                // Display the read byte:
                    System.out.print((char)b[i]);
                }
            }

            // Close the buffer:
            bin.close();
        }
        catch (IOException e)
        {
             // Do nothing.
        }
    }
}
```

Remarks

If no byte is available because the end of the stream has been reached, it returns -1.

Can throw java.io.IOException and 997bf792-c774-4384-8045-19bacb0cb4ad.

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.reset Method

Repositions the mark created by fc2bb7b1-e8ab-44ca-a9a1-362d98852478 to its last position when fc2bb7b1-e8ab-44ca-a9a1-362d98852478 was last called.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void reset() throws java.io.IOException;
```

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedInputStream.skip Method

Skips reading a specific number of bytes.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized long skip(
    long n) throws java.io.IOException;
```

Parameters

n

The number of bytes to skip.

Return Value

The actual number of skipped bytes.

See Also

Reference

BufferedInputStream Class

Concepts

BufferedInputStream Members

java.io Package

BufferedOutputStream Class

Stores the data to be written into a buffer. The data is actually written when the buffer is full or when you empty the buffer by calling the flush method.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.BufferedOutputStream
    extends java.io.FilterOutputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.OutputStream

    java.io.FilterOutputStream

      java.io.BufferedOutputStream

See Also

Concepts

BufferedOutputStream Members

java.io Package

BufferedOutputStream Members

Stores the data to be written into a buffer. The data is actually written when the buffer is full or when you empty the buffer by calling the flush method.

The following tables list the members exposed by the BufferedOutputStream type.

Public Constructors

Name	Description
BufferedOutputStream	Overloaded.

Public Fields

Name	Description
buf	Stores the array of bytes that represents the buffer data.
count	Stores the number of bytes written to the stream.
out	(inherited from FilterOutputStream)

Public Methods

Name	Description
close	Overridden. Closes the output stream.
equals	(inherited from Object)
flush	Overridden. Empties the output buffer, causing the stored data to be written to the output stream.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BufferedOutputStream Class

Concepts

java.io Package

BufferedOutputStream Fields

Public Fields

Name	Description
buf	Stores the array of bytes that represents the buffer data.
count	Stores the number of bytes written to the stream.
out	(inherited from FilterOutputStream)

See Also

Reference

BufferedOutputStream Class

Concepts

java.io Package

BufferedOutputStream.buf Field

Stores the array of bytes that represents the buffer data.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected byte[] buf;
```

See Also

Reference

BufferedOutputStream Class

Concepts

BufferedOutputStream Members

java.io Package

BufferedOutputStream.count Field

Stores the number of bytes written to the stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int count;
```

See Also

Reference

BufferedOutputStream Class

Concepts

BufferedOutputStream Members

java.io Package

BufferedOutputStream Constructor

Overload List

Name	Description
BufferedOutputStream (OutputStream)	Constructs a BufferedOutputStream with the default size.
BufferedOutputStream (OutputStream, int)	Creates a BufferedOutputStream with a specified size.

See Also

Reference

BufferedOutputStream Class

Concepts

BufferedOutputStream Members

java.io Package

BufferedOutputStream Constructor (OutputStream)

Constructs a BufferedOutputStream with the default size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.BufferedOutputStream(
    java.io.OutputStream out);
```

Parameters

out

The buffer that stores the data to be written to the output stream.

See Also
Reference
BufferedOutputStream Class
Concepts
BufferedOutputStream Members
java.io Package

BufferedOutputStream Constructor (OutputStream, Int32)

Creates a BufferedOutputStream with a specified size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.BufferedOutputStream(
    java.io.OutputStream out,
    int size);
```

Parameters

out

The buffer that stores the data to be written to the output stream.

size

The buffer size.

See Also

Reference

BufferedOutputStream Class

Concepts

BufferedOutputStream Members

java.io Package

BufferedOutputStream Methods

Public Methods

Name	Description
close	Overridden. Closes the output stream.
equals	(inherited from Object)
flush	Overridden. Empties the output buffer, causing the stored data to be written to the output stream.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BufferedOutputStream Class

Concepts

java.io Package

BufferedOutputStream.close Method

Closes the output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void close() throws java.io.IOException;
```

See Also

Reference

BufferedOutputStream Class

Concepts

BufferedOutputStream Members

java.io Package

BufferedOutputStream.flush Method

Empties the output buffer, causing the stored data to be written to the output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void flush() throws java.io.IOException;
```

See Also

Reference

BufferedOutputStream Class

Concepts

BufferedOutputStream Members

java.io Package

BufferedOutputStream.write Method

Overload List

Name	Description
BufferedOutputStream.write (int)	Writes the least significant byte of an int parameter to the stream.
BufferedOutputStream.write (byte[])	
BufferedOutputStream.write (byte[], int, int)	Writes a specific number of bytes, specified by the length of a byte array, to the output buffer.

See Also

Reference

BufferedOutputStream Class

Concepts

BufferedOutputStream Members

java.io Package

BufferedOutputStream.write Method (Int32)

Writes the least significant byte of an int parameter to the stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void write(
    int b) throws java.io.IOException;
```

Parameters

b

An int parameter.

See Also

Reference

BufferedOutputStream Class

Concepts

BufferedOutputStream Members

java.io Package

BufferedOutputStream.write Method (SByte[], Int32, Int32)

Writes a specific number of bytes, specified by the length of a byte array, to the output buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void write(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

The byte array.

off

The offset at which the writing of the data will start.

len

The number of bytes to be written.

See Also

Reference

BufferedOutputStream Class

Concepts

BufferedOutputStream Members

java.io Package

BufferedReader Class

Provides an implementation of the abstract Reader class that buffers character data from another reader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.BufferedReader
    extends java.io.Reader
```

Example

The following example demonstrates the close, mark, markSupported, read, ready, and reset methods of the BufferedReader class.

```
// bufferedreader_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            char[] arr = new char[s.length()];
            if (bufReader.ready())
            {
                bufReader.read(arr, 0, arr.length - 10);
            }

            // Call mark after having read all but the last 10
            // characters from the StringReader.
            if (bufReader.markSupported())
            {
                bufReader.mark(s.length());
            }

            // Read the rest of the data from the underlying
            // StringReader.
            if (bufReader.ready())
            {
                bufReader.read(arr, arr.length - 10, 10);
            }
            System.out.println(arr);

            // Call reset and then re-read from the underlying
            // StringReader.
            bufReader.reset();
            char[] arr2 = new char[s.length()];
            bufReader.read(arr2);
            System.out.println(arr2);

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
```

```
            bufReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal StringReader buffer.
er buffer.
*/
```

Remarks

The BufferedReader class provides functionality to read characters from a stream with buffering. It also provides mark and reset functionality for reading characters back and forth within a buffered limit from a sequential stream.

Inheritance Hierarchy

java.lang.Object

  java.io.Reader

   java.io.BufferedReader

    java.io.LineNumberReader

See Also

Concepts

BufferedReader Members

java.io Package

BufferedReader Members

Provides an implementation of the abstract Reader class that buffers character data from another reader.

The following tables list the members exposed by the BufferedReader type.

Public Constructors

Name	Description
BufferedReader	Overloaded. Initializes a new instance of a BufferedReader object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader.(inherited from Reader)

Public Methods

Name	Description
close	Overridden. Closes the stream being buffered and releases internal buffer.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Sets the mark to the next character in the stream to be read. Also specifies a limit on the number of characters that can be read before the mark becomes invalid.
markSupported	Overridden. Determines whether the current position of the stream can be marked.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the underlying stream.
readLine	Reads an entire line from the underlying stream.
ready	Overridden. Determines whether the underlying stream is ready for reading.
reset	Overridden. Resets the reader for the underlying stream such that the next character read is the character that is marked.
skip	Overridden. Skips over the next n characters in the underlying stream or until the end of the stream, whichever is less.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BufferedReader Class
Concepts
java.io Package

BufferedReader Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader. (inherited from Reader)

See Also

Reference

BufferedReader Class

Concepts

java.io Package

BufferedReader Constructor

Initializes a new instance of a BufferedReader object.

Overload List

Name	Description
BufferedReader (Reader)	Initializes a new instance of a BufferedReader object. The provided Reader object is the source of the data to be buffered.
BufferedReader (Reader, int)	Initializes a new instance of a BufferedReader object. The provided Reader object is the source of the data to be buffered. The internal buffer is initialized to the given size.

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader Constructor (Reader)

Initializes a new instance of a BufferedReader object. The provided Reader object is the source of the data to be buffered.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.BufferedReader(
    java.io.Reader in);
```

Parameters

in

A Reader object that is the source of the data to be buffered.

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader Constructor (Reader, Int32)

Initializes a new instance of a BufferedReader object. The provided Reader object is the source of the data to be buffered. The internal buffer is initialized to the given size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.BufferedReader(
    java.io.Reader in,
    int size);
```

Parameters

in

A Reader object that is the source of the data to be buffered.

size

The size of the internal buffer.

See Also
Reference
BufferedReader Class
Concepts
BufferedReader Members
java.io Package

BufferedReader Methods

Public Methods

Name	Description
close	Overridden. Closes the stream being buffered and releases internal buffer.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Sets the mark to the next character in the stream to be read. Also specifies a limit on the number of characters that can be read before the mark becomes invalid.
markSupported	Overridden. Determines whether the current position of the stream can be marked.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the underlying stream.
readLine	Reads an entire line from the underlying stream.
ready	Overridden. Determines whether the underlying stream is ready for reading.
reset	Overridden. Resets the reader for the underlying stream such that the next character read is the character that is marked.
skip	Overridden. Skips over the next n characters in the underlying stream or until the end of the stream, whichever is less.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BufferedReader Class

Concepts

java.io Package

BufferedReader.close Method

Closes the stream being buffered and releases internal buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close a BufferedReader object.

```
// bufferedreader_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            char[] arr = new char[s.length()];
            bufReader.read(arr);
            System.out.println(arr);

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
            bufReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal StringReader buffer.
*/
```

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader.mark Method

Sets the mark to the next character in the stream to be read. Also specifies a limit on the number of characters that can be read before the mark becomes invalid.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void mark(
    int readLimit) throws java.io.IOException;
```

Parameters

readLimit

The maximum number of characters that can be marked. If negative, an IllegalArgumentException is thrown.

Example

The following example demonstrates the effects that calling mark and reset have on a BufferedReader object.

```
// bufferedreader_mark.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            char[] arr = new char[s.length()];
            bufReader.read(arr, 0, arr.length - 10);

            // Call mark after having read all but the last 10
            // characters from the StringReader.
            if (bufReader.markSupported())
            {
                bufReader.mark(s.length());
            }

            // Read the rest of the data from the underlying
            // StringReader.
            bufReader.read(arr, arr.length - 10, 10);
            System.out.println(arr);

            // Call reset and then re-read from the underlying
            // StringReader.
            bufReader.reset();
            char[] arr2 = new char[s.length()];
            bufReader.read(arr2);
            System.out.println(arr2);

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
            bufReader.close();
```

```
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal StringReader buffer.
er buffer.
*/
```

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader.markSupported Method

Determines whether the current position of the stream can be marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

Return Value

Always true.

Example

The following example demonstrates the effects that calling mark and reset have on a BufferedReader object.

```
// bufferedreader_marksupported.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            char[] arr = new char[s.length()];
            bufReader.read(arr, 0, arr.length - 10);

            // Call mark after having read all but the last 10
            // characters from the StringReader.
            if (bufReader.markSupported())
            {
                bufReader.mark(s.length());
            }

            // Read the rest of the data from the underlying
            // StringReader.
            bufReader.read(arr, arr.length - 10, 10);
            System.out.println(arr);

            // Call reset and then re-read from the underlying
            // StringReader.
            bufReader.reset();
            char[] arr2 = new char[s.length()];
            bufReader.read(arr2);
            System.out.println(arr2);

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
            bufReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
```

```
        }
    }
}

/*
Output:
This is the internal StringReader buffer.
er buffer.
*/
```

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader.read Method

Reads the next character or characters from the underlying stream.

Overload List

Name	Description
BufferedReader.read ()	Reads the next character from the underlying stream.
BufferedReader.read (char[])	Reads the characters in the array starting at an offset for the length specified and stores them in the provided array of characters.
BufferedReader.read (char[], int, int)	Reads the characters from the underlying stream for the length specified, and stores them in the provided array of characters, starting at an offset.

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader.read Method ()

Reads the next character from the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

Return Value

The numeric value of the character read from the stream, or -1 if the end of the stream is reached.

Example

The following example demonstrates how to read a single character at a time from the underlying input stream.

```
// bufferedreader_read.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            for (int i = 0; i < s.length(); i++)
            {
                System.out.print((char)bufReader.read());
            }

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
            bufReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal StringReader buffer.
*/
```

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader.read Method (Char[], Int32, Int32)

Reads the characters from the underlying stream for the length specified, and stores them in the provided array of characters, starting at an offset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

The array of characters to store the characters read from the underlying stream.

off

An offset into the array of characters. This value represents the index of the first character in the array to be written. This value must be greater than zero.

len

The number of characters to be read. This value plus the offset must be less than the overall length of the array.

Return Value

The number of characters read, or -1 if the end of the stream is reached.

Example

The following example demonstrates how to read from the underlying input stream and store the data read into an array.

```
// bufferedreader_read_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            char[] arr = new char[s.length()];
            bufReader.read(arr, 0, arr.length);
            System.out.println(arr);

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
            bufReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }
}

/*
Output:
This is the internal StringReader buffer.
*/
```

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader.readLine Method

Reads an entire line from the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String readLine() throws java.io.IOException;
```

Return Value

A string representing the contents of the line read from the underlying stream. The string contains all the text in the stream up to the next carriage return (CR), line feed (LF), carriage return plus line feed (CR+LF), or the end of the stream. If no characters other than a carriage return or line feed are found, then null is returned.

Example

The following example demonstrates how to read an entire line from the underlying input stream.

```
// bufferedreader_readline.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            String str = null;
            while (true)
            {
                str = bufReader.readLine();
                if (str != null)
                {
                    System.out.println(str);
                }
                else
                    break;
            }

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
            bufReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal StringReader buffer.
```

```
null
*/
```

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader.ready Method

Determines whether the underlying stream is ready for reading.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean ready() throws java.io.IOException;
```

Return Value

true if the underlying stream is ready for reading; false otherwise.

Example

The following example demonstrates how to determine if a BufferedReader object is ready for reading.

```
// bufferedreader_ready.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            char[] arr = new char[s.length()];
            if (bufReader.ready())
            {
                bufReader.read(arr, 0, arr.length);
            }
            System.out.println(arr);

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
            bufReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal StringReader buffer.
*/
```

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

BufferedReader.reset Method

Resets the reader for the underlying stream such that the next character read is the character that is marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void reset() throws java.io.IOException;
```

Example

The following example demonstrates the effects that calling mark and reset have on a BufferedReader object.

```
// bufferedreader_reset.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            char[] arr = new char[s.length()];
            bufReader.read(arr, 0, arr.length - 10);

            // Call mark after having read all but the last 10
            // characters from the StringReader.
            if (bufReader.markSupported())
            {
                bufReader.mark(s.length());
            }

            // Read the rest of the data from the underlying
            // StringReader.
            bufReader.read(arr, arr.length - 10, 10);
            System.out.println(arr);

            // Call reset and then re-read from the underlying
            // StringReader.
            bufReader.reset();
            char[] arr2 = new char[s.length()];
            bufReader.read(arr2);
            System.out.println(arr2);

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
            bufReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
This is the internal StringReader buffer.
er buffer.
*/
```

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedReader.skip Method

Skips over the next n characters in the underlying stream or until the end of the stream, whichever is less.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long n) throws java.io.IOException;
```

Parameters

n

The number of characters to skip over. This value must be greater than zero. If this number is greater than the number of characters remaining in the stream, then only the number of remaining characters will be skipped.

Return Value

The number of characters skipped.

Example

The following example demonstrates how to skip characters when reading from the underlying input stream.

```
// bufferedreader_skip.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedReader object using
            // a StringReader.
            String s = "This is the internal StringReader buffer.";
            StringReader stringReader = new StringReader(s);
            BufferedReader bufReader = new BufferedReader(stringReader);

            // Read from the underlying StringReader.
            for (int i = 0; i < s.length(); i++)
            {
                if (bufReader.ready())
                {
                    System.out.print((char)bufReader.read());
                }

                // Skip over i characters.
                bufReader.skip(i);
            }

            // Close the BufferedReader object and the underlying
            // StringReader object.
            stringReader.close();
            bufReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
ThsseeSef??????????????????????????????
*/
```

See Also

Reference

BufferedReader Class

Concepts

BufferedReader Members

java.io Package

BufferedWriter Class

Provides an implementation of the abstract Writer class that buffers character data to another writer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.BufferedWriter
    extends java.io.Writer
```

Example

The following example demonstrates the close, flush, newLine, and write methods of the BufferedWriter class.

```
// bufferedwriter_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedWriter object using
            // a StringWriter.
            StringWriter stringWriter = new StringWriter();
            BufferedWriter bufWriter = new BufferedWriter(stringWriter);

            // Write to the underlying StringWriter.
            String s = "This is the string being written.";

            // Print out the first 12 characters.
            bufWriter.write(s, 0, 12);

            // Print a newline character.
            bufWriter.newLine();

            // Print the rest of the string.
            bufWriter.write(s, 12, s.length() - 12);
            bufWriter.flush();
            System.out.println(stringWriter.getBuffer());

            // Close the BufferedWriter object and the underlying
            // StringWriter object.
            stringWriter.close();
            bufWriter.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the
string being written.
*/
```

Inheritance Hierarchy

java.lang.Object

  java.io.Writer

    java.io.BufferedWriter

See Also

Concepts

BufferedWriter Members

java.io Package

BufferedWriter Members

Provides an implementation of the abstract Writer class that buffers character data to another writer.

The following tables list the members exposed by the BufferedWriter type.

Public Constructors

Name	Description
BufferedWriter	Overloaded. Initializes a new instance of a BufferedWriter object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer.(inherited from Writer)

Public Methods

Name	Description
close	Overridden. Closes the stream. Flushes and releases internal buffer.
equals	(inherited from Object)
flush	Overridden. Flushes the contents of the internal buffer to the underlying writer. The underlying writer is also flushed.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
newLine	Inserts a new line character into the underlying stream.
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Writes the next character or characters to the underlying stream.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BufferedWriter Class

Concepts

java.io Package

BufferedWriter Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer. (inherited from Writer)

See Also

Reference

BufferedWriter Class

Concepts

java.io Package

BufferedWriter Constructor

Initializes a new instance of a BufferedWriter object.

Overload List

Name	Description
BufferedWriter (Writer)	Initializes a new instance of a BufferedWriter object. The provided Writer object is the destination for the data being buffered.
BufferedWriter (Writer, int)	Initializes a new instance of a BufferedWriter object. The provided Writer object is the destination for the data being buffered. The internal buffer is initialized to the given size.

See Also

Reference

BufferedWriter Class

Concepts

BufferedWriter Members

java.io Package

BufferedWriter Constructor (Writer)

Initializes a new instance of a BufferedWriter object. The provided Writer object is the destination for the data being buffered.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.BufferedWriter(
    java.io.Writer out);
```

Parameters

out

The destination for the data being buffered.

See Also
Reference
BufferedWriter Class
Concepts
BufferedWriter Members
java.io Package

BufferedWriter Constructor (Writer, Int32)

Initializes a new instance of a BufferedWriter object. The provided Writer object is the destination for the data being buffered. The internal buffer is initialized to the given size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.BufferedWriter(
    java.io.Writer out,
    int size);
```

Parameters

out

The destination for the data being buffered.

size

The size of the internal buffer.

See Also

Reference

BufferedWriter Class

Concepts

BufferedWriter Members

java.io Package

BufferedWriter Methods

Public Methods

Name	Description
close	Overridden. Closes the stream. Flushes and releases internal buffer.
equals	(inherited from Object)
flush	Overridden. Flushes the contents of the internal buffer to the underlying writer. The underlying writer is also flushed.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
newLine	Inserts a new line character into the underlying stream.
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Writes the next character or characters to the underlying stream.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BufferedWriter Class

Concepts

java.io Package

BufferedWriter.close Method

Closes the stream. Flushes and releases internal buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close a BufferedWriter object.

```
// bufferedwriter_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedWriter object using
            // a StringWriter.
            StringWriter stringWriter = new StringWriter();
            BufferedWriter bufWriter = new BufferedWriter(stringWriter);

            // Write to the underlying StringWriter.
            String s = "This is the string being written.";
            bufWriter.write(s);
            bufWriter.flush();
            System.out.println(stringWriter.getBuffer());

            // Close the BufferedWriter object and the underlying
            // StringWriter object.
            stringWriter.close();
            bufWriter.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the string being written.
*/
```

See Also

Reference

BufferedWriter Class

Concepts

BufferedWriter Members

java.io Package

BufferedWriter.flush Method

Flushes the contents of the internal buffer to the underlying writer. The underlying writer is also flushed.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush() throws java.io.IOException;
```

Example

The following example demonstrates how to flush the internal buffer of a BufferedWriter object.

```
// bufferedwriter_flush.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedWriter object using
            // a StringWriter.
            StringWriter stringWriter = new StringWriter();
            BufferedWriter bufWriter = new BufferedWriter(stringWriter);

            // Write to the underlying StringWriter.
            String s = "This is the string being written.";
            bufWriter.write(s);
            bufWriter.flush();
            System.out.println(stringWriter.getBuffer());

            // Close the BufferedWriter object and the underlying
            // StringWriter object.
            stringWriter.close();
            bufWriter.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the string being written.
*/
```

See Also

Reference

BufferedWriter Class

Concepts

BufferedWriter Members

java.io Package

BufferedWriter.newLine Method

Inserts a new line character into the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void newLine() throws java.io.IOException;
```

Example

The following example demonstrates how to insert a newline character into the underlying output stream.

```
// bufferedwriter_newline.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedWriter object using
            // a StringWriter.
            StringWriter stringWriter = new StringWriter();
            BufferedWriter bufWriter = new BufferedWriter(stringWriter);

            // Write to the underlying StringWriter.
            String s = "This is the string being written.";

            // Print out the first 12 characters.
            bufWriter.write(s, 0, 12);

            // Print a newline character.
            bufWriter.newLine();

            // Print the rest of the string.
            bufWriter.write(s, 12, s.length() - 12);
            bufWriter.flush();
            System.out.println(stringWriter.getBuffer());

            // Close the BufferedWriter object and the underlying
            // StringWriter object.
            stringWriter.close();
            bufWriter.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the
string being written.
*/
```

See Also

Reference

BufferedWriter Class
Concepts
BufferedWriter Members
java.io Package

BufferedWriter.write Method

Writes the next character or characters to the underlying stream.

Overload List

Name	Description
BufferedWriter.write (char[])	Places the given array of characters into the internal buffer holding the characters.
BufferedWriter.write (int)	Writes the next character to the underlying stream.
BufferedWriter.write (String)	Places the given string into the internal buffer holding characters.
BufferedWriter.write (char[], int, int)	Writes len characters from the provided buffer, starting at an offset, to the underlying stream.
BufferedWriter.write (String, int, int)	Writes len characters from the provided string, starting at an offset, to the underlying stream.

See Also

Reference

BufferedWriter Class

Concepts

BufferedWriter Members

java.io Package

BufferedWriter.write Method (Int32)

Writes the next character to the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int b) throws java.io.IOException;
```

Parameters

b

The character to write to the underlying stream.

Example

The following example demonstrates how to write a single character at a time to the underlying output stream.

```
// bufferedwriter_write.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedWriter object using
            // a StringWriter.
            StringWriter stringWriter = new StringWriter();
            BufferedWriter bufWriter = new BufferedWriter(stringWriter);

            // Write to the underlying StringWriter.
            String s = "This is the string being written.";
            for (int i = 0; i < s.length(); i++)
            {
                bufWriter.write((char)s.charAt(i));
            }
            bufWriter.flush();
            System.out.println(stringWriter.getBuffer());

            // Close the BufferedWriter object and the underlying
            // StringWriter object.
            stringWriter.close();
            bufWriter.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the string being written.
*/
```

See Also

Reference

BufferedWriter Class

Concepts
BufferedWriter Members
java.io Package

BufferedWriter.write Method (Char[], Int32, Int32)

Writes len characters from the provided buffer, starting at an offset, to the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

An array of characters to write to the underlying stream.

off

An offset into the provided array. This value represents the index of the first character in the array to be read. This value must be greater than zero.

len

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the provided array.

Example

The following example demonstrates how to write an array of data to the underlying output stream.

```
// bufferedwriter_write_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedWriter object using
            // a StringWriter.
            StringWriter stringWriter = new StringWriter();
            BufferedWriter bufWriter = new BufferedWriter(stringWriter);

            // Write to the underlying StringWriter.
            String s = "This is the string being written.";
            char[] arr = s.toCharArray();
            bufWriter.write(arr, 0, arr.length);
            bufWriter.flush();
            System.out.println(stringWriter.getBuffer());

            // Close the BufferedWriter object and the underlying
            // StringWriter object.
            stringWriter.close();
            bufWriter.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
This is the string being written.
*/
```

See Also

Reference

BufferedWriter Class

Concepts

BufferedWriter Members

java.io Package

BufferedWriter.write Method (String, Int32, Int32)

Writes len characters from the provided string, starting at an offset, to the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str,
    int off,
    int len) throws java.io.IOException;
```

Parameters

str

A string to write to the underlying stream.

off

An offset into the provided string. This value represents the index of the first character in the string to be read. This value must be greater than zero.

len

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the provided string.

Example

The following example demonstrates how to write a string to the underlying output stream.

```
// bufferedwriter_write_3.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a BufferedWriter object using
            // a StringWriter.
            StringWriter stringWriter = new StringWriter();
            BufferedWriter bufWriter = new BufferedWriter(stringWriter);

            // Write to the underlying StringWriter.
            String s = "This is the string being written.";
            bufWriter.write(s, 0, s.length());
            bufWriter.flush();
            System.out.println(stringWriter.getBuffer());

            // Close the BufferedWriter object and the underlying
            // StringWriter object.
            stringWriter.close();
            bufWriter.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
This is the string being written.
*/
```

See Also

Reference

BufferedWriter Class

Concepts

BufferedWriter Members

java.io Package

ByteArrayInputStream Class

Contains an internal buffer that stores the bytes read from the input stream. By using a counter, it keeps track of the next byte to be read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.ByteArrayInputStream
    extends java.io.InputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

   java.io.ByteArrayInputStream

See Also

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream Members

Contains an internal buffer that stores the bytes read from the input stream. By using a counter, it keeps track of the next byte to be read.

The following tables list the members exposed by the ByteArrayInputStream type.

Public Constructors

Name	Description
ByteArrayInputStream	Overloaded.

Public Fields

Name	Description
buf	A byte array that stores the data.
count	An integer that represents the index one greater than the last valid character in the input buffer.
mark	The currently marked position in the stream.
pos	The index of the next character to read from the input stream buffer.

Public Methods

Name	Description
available	Overridden. Returns the number of bytes that can be read from the input stream without being blocked.
close	(inherited from InputStream)
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Sets the current marked position in the ByteArrayInputStream object. By default, the ByteArrayInputStream object is marked at position zero. By using this method, you can mark it at any other position.
markSupported	Overridden. Checks if the mark is supported by the input stream.
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden. Resets the buffer to the marked position. Unless the position was marked otherwise, its default position is zero. Specifying an offset in the constructor can also change the marked position.
skip	Overridden. Skips a specified amount of bytes from the input stream
toString	(inherited from InputStream)

Protected Methods

Name	Description

finalize	(inherited from Object)

See Also

Reference

ByteArrayInputStream Class

Concepts

java.io Package

ByteArrayInputStream Fields

Public Fields

Name	Description
buf	A byte array that stores the data.
count	An integer that represents the index one greater than the last valid character in the input buffer.
mark	The currently marked position in the stream.
pos	The index of the next character to read from the input stream buffer.

See Also

Reference

ByteArrayInputStream Class

Concepts

java.io Package

ByteArrayInputStream.buf Field

A byte array that stores the data.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected byte[] buf;
```

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream.count Field

An integer that represents the index one greater than the last valid character in the input buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int count;
```

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream.mark Field

The currently marked position in the stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int mark;
```

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream.pos Field

The index of the next character to read from the input stream buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int pos;
```

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream Constructor

Overload List

Name	Description
ByteArrayInputStream (byte[])	Constructs a ByteArrayInputStream object and uses the field buf as its buffer.
ByteArrayInputStream (byte[], int, int)	Constructs a ByteArrayInputStream object and uses the field buf as its buffer.

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream Constructor (SByte[ ])

Constructs a ByteArrayInputStream object and uses the field buf as its buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.ByteArrayInputStream(
    byte[] b);
```

Parameters

b

The buffer array.

Remarks

The initial pos is zero.

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream Constructor (SByte[ ], Int32, Int32)

Constructs a ByteArrayInputStream object and uses the field buf as its buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.ByteArrayInputStream(
    byte[] b,
    int off,
    int len);
```

Parameters

b

The buffer array.

off

The offset of the first byte in the buffer.

len

The maximum number of bytes to be read from the buffer.

Remarks

The initial value of pos is off.

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream Methods

Public Methods

Name	Description
available	Overridden. Returns the number of bytes that can be read from the input stream without being blocked.
close	(inherited from InputStream)
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Sets the current marked position in the ByteArrayInputStream object. By default, the ByteArrayInputStream object is marked at position zero. By using this method, you can mark it at any other position.
markSupported	Overridden. Checks if the mark is supported by the input stream.
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden. Resets the buffer to the marked position. Unless the position was marked otherwise, its default position is zero. Specifying an offset in the constructor can also change the marked position.
skip	Overridden. Skips a specified amount of bytes from the input stream
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ByteArrayInputStream Class

Concepts

java.io Package

ByteArrayInputStream.available Method

Returns the number of bytes that can be read from the input stream without being blocked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int available();
```

Return Value

The number of bytes that can be read from the ByteArrayInputStream without being blocked.

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream.mark Method

Sets the current marked position in the ByteArrayInputStream object. By default, the ByteArrayInputStream object is marked at position zero. By using this method, you can mark it at any other position.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void mark(
    int readlimit);
```

Parameters

readlimit

The maximum number of bytes that can be read before the marked position becomes invalid.

See Also
Reference
ByteArrayInputStream Class
Concepts
ByteArrayInputStream Members
java.io Package

ByteArrayInputStream.markSupported Method

Checks if the mark is supported by the input stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

Return Value

true if supported; false otherwise.

Remarks

For ByteArrayInputStream objects this method always return true.

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream.read Method

Overload List

Name	Description
ByteArrayInputStream.read ()	Reads the next byte of data from the current ByteArrayInputStream.
ByteArrayInputStream.read (byte[])	
ByteArrayInputStream.read (byte[], int, int)	Reads up to total number of bytes of data into a byte array from the current input stream object.

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream.read Method ()

Reads the next byte of data from the current ByteArrayInputStream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int read();
```

Return Value

An int value between 0 and 255 or -1 If the end of the stream has been reached.

Remarks

Might throw an **IOException**.

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream.read Method (SByte[], Int32, Int32)

Reads up to total number of bytes of data into a byte array from the current input stream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int read(
    byte[] b,
    int off,
    int len);
```

Parameters

b

The buffer of the input data.

off

The starting offset of the data.

len

The total number of bytes read into the buffer, or -1 if the end of the stream has been reached.

Return Value

If pos is greater than or equal count, -1 is returned. If len equals 0, zero is returned to indicate end of file. Otherwise, the number of bytes read is equal to the minimum of len and count-pos.

Remarks

The parameters are not validated and any runtime exception may be raised in a **System.arraycopy** call.

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream.reset Method

Resets the buffer to the marked position. Unless the position was marked otherwise, its default position is zero. Specifying an offset in the constructor can also change the marked position.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void reset();
```

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayInputStream.skip Method

Skips a specified amount of bytes from the input stream

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized long skip(
    long n);
```

Parameters

n

The number of bytes to skip.

Return Value

The number of bytes to skip.

Remarks

If the end of the stream is reached, a fewer number of bytes might be skipped. The actual number of skipped bytes is equal to the minimum of n and count-pos.

See Also

Reference

ByteArrayInputStream Class

Concepts

ByteArrayInputStream Members

java.io Package

ByteArrayOutputStream Class

Extends the **OutputStream** class in which the data is written into a byte array. It contains a buffer that automatically grows as data is written to it. It also contains methods to retrieve and convert data.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.ByteArrayOutputStream
    extends java.io.OutputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.OutputStream

   java.io.ByteArrayOutputStream

See Also

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream Members

Extends the **OutputStream** class in which the data is written into a byte array. It contains a buffer that automatically grows as data is written to it. It also contains methods to retrieve and convert data.

The following tables list the members exposed by the ByteArrayOutputStream type.

Public Constructors

Name	Description
ByteArrayOutputStream	Overloaded.

Public Fields

Name	Description
buf	The data buffer of the ByteArrayOutputStream object.
count	The counter that stores the number of bytes in the ByteArrayOutputStream buffer.

Public Methods

Name	Description
close	(inherited from OutputStream)
equals	(inherited from Object)
flush	(inherited from OutputStream)
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
reset	Resets the count field of the current ByteArrayOutputStream to zero discarding the data that exists in the output st ream.
size	Returns the number of the bytes that have been written to the current ByteArrayOutputStream object since its mo st reset or since it was created, if it has never been reset.
toByteArray	Allocates and returns new byte array and copies the buffer contents to it.
toString	Overridden. Returns the string representation of the buffer's data for the current ByteArrayOutputStream object.
write	Overloaded. Overridden.
writeTo	Writes the entire contents of the current ByteArrayOutputStream object to a specified output stream.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ByteArrayOutputStream Class
Concepts
java.io Package

ByteArrayOutputStream Fields

Public Fields

Name	Description
buf	The data buffer of the ByteArrayOutputStream object.
count	The counter that stores the number of bytes in the ByteArrayOutputStream buffer.

See Also

Reference

ByteArrayOutputStream Class

Concepts

java.io Package

ByteArrayOutputStream.buf Field

The data buffer of the ByteArrayOutputStream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected byte[] buf;
```

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.count Field

The counter that stores the number of bytes in the ByteArrayOutputStream buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int count;
```

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream Constructor

Overload List

Name	Description
ByteArrayOutputStream ()	Constructs a new ByteArrayOutputStream object with a default initial buffer size.
ByteArrayOutputStream (int)	Constructs a new ByteArrayOutputStream object with a specified initial buffer size.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream Constructor ()

Constructs a new ByteArrayOutputStream object with a default initial buffer size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.ByteArrayOutputStream();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream Constructor (Int32)

Constructs a new ByteArrayOutputStream object with a specified initial buffer size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.ByteArrayOutputStream(
    int size);
```

Parameters

size

The initial buffer size.

Remarks

Throws an **IllegalArgumentException** if the buffer size is negative.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream Methods

Public Methods

Name	Description
close	(inherited from OutputStream)
equals	(inherited from Object)
flush	(inherited from OutputStream)
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
reset	Resets the count field of the current ByteArrayOutputStream to zero discarding the data that exists in the output st ream.
size	Returns the number of the bytes that have been written to the current ByteArrayOutputStream object since its mo st reset or since it was created, if it has never been reset.
toByteArray	Allocates and returns new byte array and copies the buffer contents to it.
toString	Overridden. Returns the string representation of the buffer's data for the current ByteArrayOutputStream object.
write	Overloaded. Overridden.
writeTo	Writes the entire contents of the current ByteArrayOutputStream object to a specified output stream.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ByteArrayOutputStream Class

Concepts

java.io Package

ByteArrayOutputStream.reset Method

Resets the count field of the current ByteArrayOutputStream to zero discarding the data that exists in the output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void reset();
```

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.size Method

Returns the number of the bytes that have been written to the current ByteArrayOutputStream object since its most reset or since it was created, if it has never been reset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

Return Value

The buffer size of the ByteArrayOutputStream.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.toByteArray Method

Allocates and returns new byte array and copies the buffer contents to it.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized byte[] toByteArray();
```

Return Value

A byte array that contains the buffer data of the current ByteArrayOutputStream object.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.toString Method

Overload List

Name	Description
ByteArrayOutputStream.toString (int)	Returns the string that represents the current contents of the ByteArrayOutputStream object. The size of the string is the same as that of the buffer. The characters of the resulting string are constructed from the elements of the byte array.
ByteArrayOutputStream.toString (String)	Returns the string representation of the buffer's contents for the current ByteArrayOutputStream object. The string conversion is performed using a specified character encoding.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.toString Method (Int32)

Returns the string that represents the current contents of the ByteArrayOutputStream object. The size of the string is the same as that of the buffer. The characters of the resulting string are constructed from the elements of the byte array.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString(
    int hibyte);
```

Parameters

hibyte

The high byte of the resulting character.

Return Value

The string that represents the current contents of the ByteArrayOutputStream object.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.toString Method (String)

Returns the string representation of the buffer's contents for the current ByteArrayOutputStream object. The string conversion is performed using a specified character encoding.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString(
    java.lang.String enc) throws java.io.UnsupportedEncodingException;
```

Parameters

enc

The character encoding.

Return Value

The string representation of the buffer's data using the specified character encoding.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.toString Method (J#)

Returns the string representation of the buffer's data for the current ByteArrayOutputStream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The string representation of the buffer's data.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.write Method

Overload List

Name	Description
ByteArrayOutputStream.write (int)	Writes a byte to the current ByteArrayOutputStream object.
ByteArrayOutputStream.write (byte[])	
ByteArrayOutputStream.write (byte[], int, int)	Writes a specific number of bytes from a byte array to the current ByteArrayOutputStream object starting at a specified offset.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.write Method (Int32)

Writes a byte to the current ByteArrayOutputStream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void write(
    int b);
```

Parameters

b

The byte to be written.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.write Method (SByte[], Int32, Int32)

Writes a specific number of bytes from a byte array to the current ByteArrayOutputStream object starting at a specified offset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void write(
    byte[] b,
    int off,
    int len);
```

Parameters

b

The data byte array.

off

The offset to start at.

len

The number of bytes to be written.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

ByteArrayOutputStream.writeTo Method

Writes the entire contents of the current ByteArrayOutputStream object to a specified output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void writeTo(
    java.io.OutputStream out) throws java.io.IOException;
```

Parameters

out

The output stream.

See Also

Reference

ByteArrayOutputStream Class

Concepts

ByteArrayOutputStream Members

java.io Package

CharArrayReader Class

Provides an implementation of the abstract Reader class specific to reading arrays of characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.CharArrayReader
    extends java.io.Reader
```

Example

The following example demonstrates the close, mark, markSupported, read, and reset methods of the CharArrayReader class.

```
// chararrayreader_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new CharArrayReader.
            char[] arr = new char[] { 'H', 'e', 'l', 'l', 'o' };
            CharArrayReader reader = new CharArrayReader(arr);

            // Read from the underlying array one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the array.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the CharArrayReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the CharArrayReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
```

```
Output:
The next letter read after reset is: e
*/
```

Inheritance Hierarchy

java.lang.Object

  java.io.Reader

   java.io.CharArrayReader

See Also

Concepts

CharArrayReader Members

java.io Package

CharArrayReader Members

Provides an implementation of the abstract Reader class specific to reading arrays of characters.

The following tables list the members exposed by the CharArrayReader type.

Public Constructors

Name	Description
CharArrayReader	Overloaded. Initializes a new instance of a CharArrayReader object.

Public Fields

Name	Description
buf	The array of characters being read.
count	The number of characters in the array buf.
lock	The object used to synchronize operations on the reader.(inherited from Reader)
markedPos	The currently marked position in the array of characters.
pos	The index of the next character to be read from the array of characters.

Public Methods

Name	Description
close	Overridden. Closes the reader once the array of characters has been processed.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Sets the mark to the next character in the array of characters to be read.
markSupported	Overridden. Determines whether the current position of the array of characters can be marked.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters in the array.
ready	Overridden. Determines whether the array of characters is ready for reading.
reset	Overridden. Resets the reader for the array of characters such that the next character read is the character that is marked.
skip	Overridden. Skips over the next n characters in the array of characters or until the end of the array, whichever is less.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description

finalize	(inherited from Object)

See Also

Reference

CharArrayReader Class

Concepts

java.io Package

CharArrayReader Fields

Public Fields

Name	Description
buf	The array of characters being read.
count	The number of characters in the array buf.
lock	The object used to synchronize operations on the reader. (inherited from Reader)
markedPos	The currently marked position in the array of characters.
pos	The index of the next character to be read from the array of characters.

See Also

Reference

CharArrayReader Class

Concepts

java.io Package

CharArrayReader.buf Field

The array of characters being read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected char[] buf;
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.count Field

The number of characters in the array buf.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int count;
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.markedPos Field

The currently marked position in the array of characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int markedPos;
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.pos Field

The index of the next character to be read from the array of characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int pos;
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader Constructor

Initializes a new instance of a CharArrayReader object.

Overload List

Name	Description
CharArrayReader (char[])	Initializes a new instance of a CharArrayReader object using the given array of characters.
CharArrayReader (char[], int, int)	Initializes a new instance of a CharArrayReader object using the given array of characters, off set, and length.

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader Constructor (Char[ ])

Initializes a new instance of a CharArrayReader object using the given array of characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.CharArrayReader(
    char[] b);
```

Parameters

b

The array of characters to be read.

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader Constructor (Char[ ], Int32, Int32)

Initializes a new instance of a CharArrayReader object using the given array of characters, offset, and length.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.CharArrayReader(
    char[] b,
    int off,
    int len);
```

Parameters

b

The array of characters to be read.

off

An offset into the array of characters. This value represents the index of the first character in the array to be read. This value must be greater than zero.

len

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the array.

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader Methods

Public Methods

Name	Description
close	Overridden. Closes the reader once the array of characters has been processed.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Sets the mark to the next character in the array of characters to be read.
markSupported	Overridden. Determines whether the current position of the array of characters can be marked.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters in the array.
ready	Overridden. Determines whether the array of characters is ready for reading.
reset	Overridden. Resets the reader for the array of characters such that the next character read is the character that is marked.
skip	Overridden. Skips over the next n characters in the array of characters or until the end of the array, whichever is less.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

CharArrayReader Class

Concepts

java.io Package

CharArrayReader.close Method

Closes the reader once the array of characters has been processed.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close();
```

Example

The following example demonstrates how to close a CharArrayReader object.

```
// chararrayreader_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new CharArrayReader.
            char[] arr = new char[] { 'H', 'e', 'l', 'l', 'o' };
            CharArrayReader reader = new CharArrayReader(arr);

            // Read from the underlying array.
            char[] newArr = new char[arr.length];
            reader.read(newArr);
            System.out.println("The new array contains: " +
                newArr);

            // Close the CharArrayReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The new array contains: Hello
*/
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.mark Method

Sets the mark to the next character in the array of characters to be read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void mark(
    int readlimit) throws java.io.IOException;
```

Parameters

readlimit

Ignored.

Example

The following example demonstrates the effect that the mark method has when you call reset.

```
// chararrayreader_mark.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new CharArrayReader.
            char[] arr = new char[] { 'H', 'e', 'l', 'l', 'o' };
            CharArrayReader reader = new CharArrayReader(arr);

            // Read from the underlying array one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the array.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the CharArrayReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the CharArrayReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }
}

/*
Output:
The next letter read after reset is: e
*/
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.markSupported Method

Determines whether the current position of the array of characters can be marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

Return Value

Always true.

Example

The following example demonstrates the effect that the mark method has when you call reset.

```
// chararrayreader_marksupported.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new CharArrayReader.
            char[] arr = new char[] { 'H', 'e', 'l', 'l', 'o' };
            CharArrayReader reader = new CharArrayReader(arr);

            // Read from the underlying array one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the array.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the CharArrayReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the CharArrayReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
The next letter read after reset is: e
*/
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.read Method

Reads the next character or characters in the array.

Overload List

Name	Description
CharArrayReader.read ()	Reads the next character in the array.
CharArrayReader.read (char[])	Reads the characters in the array starting at an offset for the length specified and stores them in the provided array of characters.
CharArrayReader.read (char[], int, int)	Reads the characters in the underlying array for the length specified and stores them in the provided array of characters, starting at an offset.

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.read Method ()

Reads the next character in the array.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

Return Value

The numeric value of the character read from the underlying character array, or -1 if there are no more characters to read.

Example

The following example demonstrates how to read one character at a time from the CharArrayReader buffer.

```
// chararrayreader_read.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new CharArrayReader.
            char[] arr = new char[] { 'H', 'e', 'l', 'l', 'o' };
            CharArrayReader reader = new CharArrayReader(arr);

            // Read from the underlying array.
            for (int i = 0; i < arr.length; i++)
            {
                if (reader.ready())
                {
                    char charRead = (char)reader.read();
                    System.out.println(charRead);
                }
                else
                {
                    break;
                }
            }

            // Close the CharArrayReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
H
e
l
l
o
*/
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.read Method (Char[], Int32, Int32)

Reads the characters in the underlying array for the length specified and stores them in the provided array of characters, starting at an offset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

The array of characters to store the characters read.

off

An offset into the array of characters. This value represents the index of the first character in the array to be written. This value must be greater than zero.

len

The number of characters to be read from the underlying array. This value plus the offset must be less than the overall length of the array.

Return Value

The number of characters read.

Example

The following example demonstrates how to read from the CharArrayReader buffer and populate an already existing array with the contents just read.

```
// chararrayreader_read_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new CharArrayReader.
            char[] arr = new char[] { 'H', 'e', 'l', 'l', 'o' };
            CharArrayReader reader = new CharArrayReader(arr);

            // Read from the underlying array.
            char[] newArr = new char[arr.length + 6];
            newArr[6] = 'W';
            newArr[7] = 'o';
            newArr[8] = 'r';
            newArr[9] = 'l';
            newArr[10] = 'd';

            reader.read(newArr, 0, arr.length);
            System.out.println("The new array contains: " +
                newArr);

            // Close the CharArrayReader.
            reader.close();
```

```
            }
            catch (IOException ex)
            {
                System.out.println(ex.toString());
            }
        }
    }

    /*
    Output:
    The new array contains: Hello World
    */
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.ready Method

Determines whether the array of characters is ready for reading.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean ready() throws java.io.IOException;
```

Return Value

true if the array of characters is ready for reading; false otherwise.

Example

The following example demonstrates how to determine if a CharArrayReader object is ready for reading.

```
// chararrayreader_ready.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new CharArrayReader.
            char[] arr = new char[] { 'H', 'e', 'l', 'l', 'o' };
            CharArrayReader reader = new CharArrayReader(arr);

            // Read from the underlying array.
            for (int i = 0; i < arr.length; i++)
            {
                if (reader.ready())
                {
                    char charRead = (char)reader.read();
                    System.out.println(charRead);
                }
                else
                {
                    break;
                }
            }

            // Close the CharArrayReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
H
e
l
l
o
*/
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.reset Method

Resets the reader for the array of characters such that the next character read is the character that is marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void reset() throws java.io.IOException;
```

Example

The following example demonstrates the effect that the mark method has when you call reset.

```
// chararrayreader_reset.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new CharArrayReader.
            char[] arr = new char[] { 'H', 'e', 'l', 'l', 'o' };
            CharArrayReader reader = new CharArrayReader(arr);

            // Read from the underlying array one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the array.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the CharArrayReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the CharArrayReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
```

```
  The next letter read after reset is: e
  */
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayReader.skip Method

Skips over the next n characters in the array of characters or until the end of the array, whichever is less.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long n) throws java.io.IOException;
```

Parameters

n

The number of characters to skip over. This value must be greater than zero. If this number is greater than the number of characters remaining in the array, then only the number of remaining characters will be skipped.

Return Value

The number of characters skipped.

Example

The following example demonstrates how to skip over character while you are performing a read operation on a CharArrayReader object.

```
// chararrayreader_skip.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new CharArrayReader.
            char[] arr = new char[] { 'H', 'e', 'l', 'l', 'o' };
            CharArrayReader reader = new CharArrayReader(arr);

            // Read from the underlying array.
            for (int i = 0; i < arr.length; i++)
            {
                if (reader.ready())
                {
                    char charRead = (char)reader.read();
                    System.out.println(charRead);

                    // Skip over one character.
                    long num = reader.skip(1);
                    System.out.println("skipping " + num);
                }
                else
                {
                    break;
                }
            }

            // Close the CharArrayReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }
}

/*
Output:
H
skipping 1
l
skipping 1
o
skipping 0
?
skipping 0
?
skipping 0
*/
```

See Also

Reference

CharArrayReader Class

Concepts

CharArrayReader Members

java.io Package

CharArrayWriter Class

Provides an implementation of the abstract Writer class specific to writing into an array of characters. The array can be used to retrieve the written character data as a character array or a string or to write the character buffer to another instance of a Writer object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.CharArrayWriter
    extends java.io.Writer
```

Example

The following example demonstrates the close, flush, reset, toCharArray, write, and writeTo methods of the CharArrayWriter class.

```
// chararraywriter_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a CharArrayWriter object that can hold 11
            // characters.
            CharArrayWriter writer = new CharArrayWriter(11);

            // Write some data to the CharArrayWriter object.
            // Start with the 7th character and write to the end.
            String s = "Hello World";
            writer.write("Hello World", 6, s.length() - 6);

            // Write the contents of the CharArrayWriter object to
            // a String Writer object.
            StringWriter sWriter = new StringWriter();
            writer.writeTo(sWriter);

            // Print out the contents of the CharArrayWriter buffer.
            System.out.println("The CharArrayWriter buffer contains: " +
                writer.toString());

            // Print out the contents of the StringWriter buffer.
            System.out.println("The StringWriter buffer contains: " +
                sWriter.toString());

            // Now flush the buffer we just populated.
            writer.flush();

            // Print out the contents of the CharArrayWriter buffer.
            System.out.println("The post-flush CharArrayWriter buffer " +
                "contains: " + writer.toCharArray());

            // Now reset the buffer we just populated.
            writer.reset();

            // Print out the contents of the CharArrayWriter buffer.
            System.out.println("The post-reset CharArrayWriter buffer " +
                "contains: " + writer.toCharArray());
```

```
            // Close the CharArrayWriter and StringWriter buffers.
            writer.close();
            sWriter.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The CharArrayWriter buffer contains: World
The StringWriter buffer contains: World
The post-flush CharArrayWriter buffer contains: World
The post-reset CharArrayWriter buffer contains:
*/
```

Inheritance Hierarchy

java.lang.Object

  java.io.Writer

  java.io.CharArrayWriter

See Also

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter Members

Provides an implementation of the abstract Writer class specific to writing into an array of characters. The array can be used to retrieve the written character data as a character array or a string or to write the character buffer to another instance of a Writer object.

The following tables list the members exposed by the CharArrayWriter type.

Public Constructors

Name	Description
CharArrayWriter	Overloaded. Initializes a new instance of a CharArrayWriter object.

Public Fields

Name	Description
buf	An internal buffer that stores the array of characters being written.
count	The number of characters stored in the internal array buf.
lock	The object used to synchronize operations on the writer.(inherited from Writer)

Public Methods

Name	Description
close	Overridden. Closes the writer once the array of characters has been processed.
equals	(inherited from Object)
flush	Overridden. Flushes the buffer representing the array of characters to be written.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
reset	Resets the buffer holding the array of characters. All data currently in the buffer is discarded.
size	Returns the number of character in the buffer holding the array of characters since the object was created or last r eset.
toCharArray	Populates an array of characters with the data held in the internal buffer.
toString	Overridden. Displays a human readable representation of a CharArrayWriter object. The contents of the buffer hol ding the array of characters is returned.
write	Overloaded. Writes the next character or characters into the array.
writeTo	Writes the contents of this CharArrayWriter object to another Writer object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

CharArrayWriter Class

Concepts

java.io Package

CharArrayWriter Fields

Public Fields

Name	Description
buf	An internal buffer that stores the array of characters being written.
count	The number of characters stored in the internal array buf.
lock	The object used to synchronize operations on the writer. (inherited from Writer)

See Also

Reference

CharArrayWriter Class

Concepts

java.io Package

CharArrayWriter.buf Field

An internal buffer that stores the array of characters being written.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected char[] buf;
```

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter.count Field

The number of characters stored in the internal array buf.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int count;
```

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter Constructor

Initializes a new instance of a CharArrayWriter object.

Overload List

Name	Description
CharArrayWriter ()	Initializes a new instance of a CharArrayWriter object.
CharArrayWriter (int)	Initializes a new instance of a CharArrayWriter object. The buffer containing the array of characters is initialized to the given size.

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter Constructor ()

Initializes a new instance of a CharArrayWriter object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.CharArrayWriter();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter Constructor (Int32)

Initializes a new instance of a CharArrayWriter object. The buffer containing the array of characters is initialized to the given size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.CharArrayWriter(
    int size);
```

Parameters

size

The length for the buffer that will store the array of characters. This value must be greater than zero.

See Also
Reference
CharArrayWriter Class
Concepts
CharArrayWriter Members
java.io Package

CharArrayWriter Methods

Public Methods

Name	Description
close	Overridden. Closes the writer once the array of characters has been processed.
equals	(inherited from Object)
flush	Overridden. Flushes the buffer representing the array of characters to be written.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
reset	Resets the buffer holding the array of characters. All data currently in the buffer is discarded.
size	Returns the number of character in the buffer holding the array of characters since the object was created or last reset.
toCharArray	Populates an array of characters with the data held in the internal buffer.
toString	Overridden. Displays a human readable representation of a CharArrayWriter object. The contents of the buffer holding the array of characters is returned.
write	Overloaded. Writes the next character or characters into the array.
writeTo	Writes the contents of this CharArrayWriter object to another Writer object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

CharArrayWriter Class

Concepts

java.io Package

CharArrayWriter.close Method

Closes the writer once the array of characters has been processed.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close();
```

Example

The following example demonstrates how to close a CharArrayWriter object.

```
// chararraywriter_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a CharArrayWriter object that can hold 11
        // characters.
        CharArrayWriter writer = new CharArrayWriter(11);

        // Write some data to the CharArrayWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the CharArrayWriter buffer.
        System.out.println("The CharArrayWriter buffer contains: " +
            writer.toCharArray());

        // Close the CharArrayWriter buffer.
        writer.close();
    }
}

/*
Output:
The CharArrayWriter buffer contains: Hello World
*/
```

Remarks

This method does nothing for CharArrayWriter objects.

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter.flush Method

Flushes the buffer representing the array of characters to be written.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush();
```

Example

The following example demonstrates that the flush method for CharArrayWriter objects has no effect on the underlying buffer.

```
// chararraywriter_flush.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a CharArrayWriter object that can hold 11
        // characters.
        CharArrayWriter writer = new CharArrayWriter(11);

        // Write some data to the CharArrayWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the CharArrayWriter buffer.
        System.out.println("The CharArrayWriter buffer contains: " +
            writer.toCharArray());

        // Now flush the buffer we just populated.
        writer.flush();

        // Print out the contents of the CharArrayWriter buffer.
        System.out.println("The post-flush CharArrayWriter buffer " +
            "contains: " + writer.toCharArray());

        // Close the CharArrayWriter buffer.
        writer.close();
    }
}

/*
Output:
The CharArrayWriter buffer contains: Hello World
The post-flush CharArrayWriter buffer contains: Hello World
*/
```

Remarks

This method does nothing for CharArrayWriter objects.

See Also

Reference

CharArrayWriter Class

Concepts
CharArrayWriter Members
java.io Package

CharArrayWriter.reset Method

Resets the buffer holding the array of characters. All data currently in the buffer is discarded.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void reset();
```

Example

The following example demonstrates the effect of calling reset on a CharArrayWriter object.

```
// chararraywriter_reset.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a CharArrayWriter object that can hold 11
        // characters.
        CharArrayWriter writer = new CharArrayWriter(11);

        // Write some data to the CharArrayWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the CharArrayWriter buffer.
        System.out.println("The CharArrayWriter buffer contains: " +
            writer.toCharArray());

        // Now reset the buffer we just populated.
        writer.reset();

        // Print out the contents of the CharArrayWriter buffer.
        System.out.println("The post-reset CharArrayWriter buffer " +
            "contains: " + writer.toCharArray());

        // Close the CharArrayWriter buffer.
        writer.close();
    }
}

/*
Output:
The CharArrayWriter buffer contains: Hello World
The post-reset CharArrayWriter buffer contains:
*/
```

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter.size Method

Returns the number of character in the buffer holding the array of characters since the object was created or last reset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

Return Value

The number of character in the buffer holding the array of characters since the object was created or last reset.

Example

The following example prints out the size of a CharArrayWriter buffer.

```
// chararraywriter_size.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a CharArrayWriter object that can hold 11
            // characters.
            CharArrayWriter writer = new CharArrayWriter(11);

            // Fill the buffer with some data.
            writer.write("E=mc2");

            // Display the size of the CharArrayWriter buffer.
            System.out.println("The size of the bufer is " +
                writer.size());

            // Close the CharArrayWriter buffer.
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The size of the bufer is 5
*/
```

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter.toCharArray Method

Populates an array of characters with the data held in the internal buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public char[] toCharArray();
```

Return Value

An array of characters holding the data contained in the internal buffer.

Example

The following example shows how to display the contents of the CharArrayWriter buffer.

```
// chararraywriter_tochararray.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a CharArrayWriter object that can hold 11
        // characters.
        CharArrayWriter writer = new CharArrayWriter(11);

        // Write some data to the CharArrayWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the CharArrayWriter buffer.
        System.out.println("The CharArrayWriter buffer contains: " +
            writer.toCharArray());

        // Close the CharArrayWriter buffer.
        writer.close();
    }
}

/*
Output:
The CharArrayWriter buffer contains: Hello World
*/
```

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter.toString Method

Displays a human readable representation of a CharArrayWriter object. The contents of the buffer holding the array of characters is returned.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

A human readable representation of a CharArrayWriter object. The contents of the buffer holding the array of characters is returned.

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter.write Method

Writes the next character or characters into the array.

Overload List

Name	Description
CharArrayWriter.write (char[])	Places the given array of characters into the internal buffer holding the characters.
CharArrayWriter.write (int)	Places a character into the next position of the array of characters.
CharArrayWriter.write (String)	Places the given string into the internal buffer holding characters.
CharArrayWriter.write (char[], int, int)	Places the given array of characters, starting from the offset and for the length specified, into the internal buffer holding the array of characters.
CharArrayWriter.write (String, int, int)	Places the given string, starting from the offset and for the length specified, into the internal buffer holding the array of characters.

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter.write Method (Int32)

Places a character into the next position of the array of characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int b);
```

Parameters

b

The character to be placed into the next position of the array.

Example

The following example shows how to write one character at a time to a CharArrayWriter object.

```
// chararraywriter_write.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a CharArrayWriter object that can hold 11
        // characters.
        CharArrayWriter writer = new CharArrayWriter(11);

        // Write some data to the CharArrayWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the CharArrayWriter buffer.
        System.out.println("The CharArrayWriter buffer contains: " +
            writer.toCharArray());

        // Close the CharArrayWriter buffer.
        writer.close();
    }
}

/*
Output:
The CharArrayWriter buffer contains: Hello World
*/
```

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter.write Method (Char[], Int32, Int32)

Places the given array of characters, starting from the offset and for the length specified, into the internal buffer holding the array of characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    char[] b,
    int off,
    int len);
```

Parameters

b

The array of characters to be written to the internal buffer.

off

An offset into the array of characters. This value represents the index of the first character in the array to be written to the internal buffer. This value must be greater than zero.

len

The number of characters to be written to the internal buffer starting from the offset. This value plus the offset must be less than the overall length of the array.

Example

The following example shows how to write an array of characters to an underlying CharArrayWriter buffer.

```
// chararraywriter_write_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a CharArrayWriter object that can hold 11
        // characters.
        CharArrayWriter writer = new CharArrayWriter(11);

        // Write some data to the CharArrayWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        // Start with the 7th character and write to the end.
        writer.write(arr, 6, arr.length - 6);

        // Print out the contents of the CharArrayWriter buffer.
        System.out.println("The CharArrayWriter buffer contains: " +
            writer.toCharArray());

        // Close the CharArrayWriter buffer.
        writer.close();
    }
}

/*
Output:
The CharArrayWriter buffer contains: World
*/
```

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharArrayWriter.write Method (String, Int32, Int32)

Places the given string, starting from the offset and for the length specified, into the internal buffer holding the array of characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str,
    int off,
    int len);
```

Parameters

str

The string to be written to the internal buffer.

off

An offset into the string. This value represents the index of the first character in the string to be written to the internal buffer. This value must be greater than zero.

len

The number of characters to be written to the internal buffer starting from the offset. This value plus the offset must be less than the overall length of the array.

Example

The following example shows how to write a string to an underlying CharArrayWriter buffer.

```
// chararraywriter_write_3.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a CharArrayWriter object that can hold 11
        // characters.
        CharArrayWriter writer = new CharArrayWriter(11);

        // Write some data to the CharArrayWriter object.
        // Start with the 7th character and write to the end.
        String s = "Hello World";
        writer.write("Hello World", 6, s.length() - 6);

        // Print out the contents of the CharArrayWriter buffer.
        System.out.println("The CharArrayWriter buffer contains: " +
            writer.toCharArray());

        // Close the CharArrayWriter buffer.
        writer.close();
    }
}

/*
Output:
The CharArrayWriter buffer contains: World
*/
```

See Also

Reference
CharArrayWriter Class
Concepts
CharArrayWriter Members
java.io Package

CharArrayWriter.writeTo Method

Writes the contents of this CharArrayWriter object to another Writer object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeTo(
    java.io.Writer out) throws java.io.IOException;
```

Parameters

out

The Writer object to write to.

Example

The following example shows how to write the contents of a CharArrayWriter object to a StringWriter object.

```
// chararraywriter_writeto.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a CharArrayWriter object that can hold 11
            // characters.
            CharArrayWriter writer = new CharArrayWriter(11);

            // Write some data to the CharArrayWriter object.
            // Start with the 7th character and write to the end.
            String s = "Hello World";
            writer.write("Hello World", 6, s.length() - 6);

            // Write the contents of the CharArrayWriter object to
            // a String Writer object.
            StringWriter sWriter = new StringWriter();
            writer.writeTo(sWriter);

            // Print out the contents of the StringWriter buffer.
            System.out.println("The StringWriter buffer contains: " +
                sWriter.toString());

            // Close the CharArrayWriter and StringWriter buffers.
            writer.close();
            sWriter.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The StringWriter buffer contains: World
*/
```

See Also

Reference

CharArrayWriter Class

Concepts

CharArrayWriter Members

java.io Package

CharConversionException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.CharConversionException
    extends java.io.IOException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.CharConversionException

See Also

Concepts

CharConversionException Members

java.io Package

CharConversionException Members

The following tables list the members exposed by the CharConversionException type.

Public Constructors

Name	Description
CharConversionException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

CharConversionException Class

Concepts

java.io Package

CharConversionException Constructor

Overload List

Name	Description
CharConversionException ()	
CharConversionException (String)	
CharConversionException (SerializationInfo, StreamingContext)	
CharConversionException (String, Exception)	

See Also

Reference

CharConversionException Class

Concepts

CharConversionException Members

java.io Package

CharConversionException Constructor ()

Initializes a new instance of the CharConversionException Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.CharConversionException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

CharConversionException Class

Concepts

CharConversionException Members

java.io Package

CharConversionException Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.CharConversionException(
    java.lang.String s);
```

Parameters

s

See Also

Reference

CharConversionException Class

Concepts

CharConversionException Members

java.io Package

CharConversionException Constructor (SerializationInfo, StreamingContext)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.CharConversionException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

CharConversionException Class

Concepts

CharConversionException Members

java.io Package

CharConversionException Constructor (String. Exception)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.CharConversionException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also
Reference
CharConversionException Class
Concepts
CharConversionException Members
java.io Package

CharConversionException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

CharConversionException Class

Concepts

java.io Package

CharConversionException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

CharConversionException Class

Concepts

java.io Package

DataInput Interface

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public interface java.io.DataInput
```

See Also

Concepts

DataInput Members

java.io Package

DataInput Members

The following tables list the members exposed by the DataInput type.

Public Methods

Name	Description
readBoolean	
readByte	
readChar	
readDouble	
readFloat	
readFully	Overloaded.
readInt	
readLine	
readLong	
readShort	
readUnsignedByte	
readUnsignedShort	
readUTF	
skipBytes	

See Also

Reference

DataInput Interface

Concepts

java.io Package

DataInput Methods

Public Methods

Name	Description
readBoolean	
readByte	
readChar	
readDouble	
readFloat	
readFully	Overloaded.
readInt	
readLine	
readLong	
readShort	
readUnsignedByte	
readUnsignedShort	
readUTF	
skipBytes	

See Also

Reference

DataInput Interface

Concepts

java.io Package

DataInput.readBoolean Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean readBoolean() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readByte Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract byte readByte() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readChar Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract char readChar() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readDouble Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract double readDouble() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readFloat Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract float readFloat() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readFully Method

Overload List

Name	Description
DataInput.readFully (byte[])	
DataInput.readFully (byte[], int, int)	

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readFully Method (SByte[])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void readFully(
    byte[] buffer) throws java.io.IOException;
```

Parameters

buffer

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readFully Method (SByte [], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void readFully(
    byte[] buffer,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buffer

offset

count

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readInt Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int readInt() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readLine Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String readLine() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readLong Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract long readLong() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readShort Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract short readShort() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readUnsignedByte Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int readUnsignedByte() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readUnsignedShort Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int readUnsignedShort() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.readUTF Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String readUTF() throws java.io.IOException;
```

See Also

Reference

DataInput Interface

Concepts

DataInput Members

java.io Package

DataInput.skipBytes Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int skipBytes(
    int count) throws java.io.IOException;
```

Parameters

count

See Also
Reference
DataInput Interface
Concepts
DataInput Members
java.io Package

DataInputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.DataInputStream
    extends java.io.FilterInputStream
    implements java.io.DataInput
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

    java.io.FilterInputStream

      java.io.DataInputStream

See Also

Concepts

DataInputStream Members

java.io Package

DataInputStream Members

The following tables list the members exposed by the DataInputStream type.

Public Constructors

Name	Description
DataInputStream	

Public Fields

Name	Description
in	(inherited from FilterInputStream)

Public Methods

Name	Description
available	(inherited from FilterInputStream)
close	(inherited from FilterInputStream)
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from FilterInputStream)
markSupported	(inherited from FilterInputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
readBoolean	
readByte	
readChar	
readDouble	
readFloat	
readFully	Overloaded.
readInt	
readLine	
readLong	

readShort	
readUnsignedByte	
readUnsignedShort	
readUTF	Overloaded.
reset	(inherited from FilterInputStream)
skip	(inherited from FilterInputStream)
skipBytes	
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DataInputStream Class

Concepts

java.io Package

DataInputStream Fields

Public Fields

Name	Description
in	(inherited from FilterInputStream)

See Also

Reference

DataInputStream Class

Concepts

java.io Package

DataInputStream Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.DataInputStream(
    java.io.InputStream in);
```

Parameters

in

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream Methods

Public Methods

Name	Description
available	(inherited from FilterInputStream)
close	(inherited from FilterInputStream)
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from FilterInputStream)
markSupported	(inherited from FilterInputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
readBoolean	
readByte	
readChar	
readDouble	
readFloat	
readFully	Overloaded.
readInt	
readLine	
readLong	
readShort	
readUnsignedByte	
readUnsignedShort	
readUTF	Overloaded.
reset	(inherited from FilterInputStream)
skip	(inherited from FilterInputStream)

skipBytes	
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DataInputStream Class

Concepts

java.io Package

DataInputStream.read Method

Overload List

Name	Description
DataInputStream.read ()	
DataInputStream.read (byte[])	
DataInputStream.read (byte[], int, int)	

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int read() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.read Method (SByte[])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int read(
    byte[] b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.read Method (SByte[], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int read(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readBoolean Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final boolean readBoolean() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readByte Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final byte readByte() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readChar Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final char readChar() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readDouble Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final double readDouble() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readFloat Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final float readFloat() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readFully Method

Overload List

Name	Description
DataInputStream.readFully (byte[])	
DataInputStream.readFully (byte[], int, int)	

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readFully Method (SByte [ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void readFully(
    byte[] b) throws java.io.IOException;
```

Parameters

b

See Also
Reference
DataInputStream Class
Concepts
DataInputStream Members
java.io Package

DataInputStream.readFully Method (SByte [ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void readFully(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readInt Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int readInt() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readLine Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String readLine() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readLong Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final long readLong() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readShort Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final short readShort() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readUnsignedByte Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int readUnsignedByte() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readUnsignedShort Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int readUnsignedShort() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readUTF Method

Overload List

Name	Description
DataInputStream.readUTF ()	
DataInputStream.readUTF (DataInput)	

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readUTF Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String readUTF() throws java.io.IOException;
```

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.readUTF Method (DataInput)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.String readUTF(
    java.io.DataInput din) throws java.io.IOException;
```

Parameters

din

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataInputStream.skipBytes Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int skipBytes(
    int n) throws java.io.IOException;
```

Parameters

n

See Also

Reference

DataInputStream Class

Concepts

DataInputStream Members

java.io Package

DataOutput Interface

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public interface java.io.DataOutput
```

See Also

Concepts

DataOutput Members

java.io Package

DataOutput Members

The following tables list the members exposed by the DataOutput type.

Public Methods

Name	Description
write	Overloaded.
writeBoolean	
writeByte	
writeBytes	
writeChar	
writeChars	
writeDouble	
writeFloat	
writeInt	
writeLong	
writeShort	
writeUTF	

See Also

Reference

DataOutput Interface

Concepts

java.io Package

DataOutput Methods

Public Methods

Name	Description
write	Overloaded.
writeBoolean	
writeByte	
writeBytes	
writeChar	
writeChars	
writeDouble	
writeFloat	
writeInt	
writeLong	
writeShort	
writeUTF	

See Also

Reference

DataOutput Interface

Concepts

java.io Package

DataOutput.write Method

Overload List

Name	Description
DataOutput.write (int)	
DataOutput.write (byte[])	
DataOutput.write (byte[], int, int)	

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void write(
    int oneByte) throws java.io.IOException;
```

Parameters

oneByte

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.write Method (SByte[])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void write(
    byte[] buffer) throws java.io.IOException;
```

Parameters

buffer

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.write Method (SByte[], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void write(
    byte[] buffer,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buffer

offset

count

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeBoolean Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeBoolean(
    boolean val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeByte Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeByte(
    int val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeBytes Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeBytes(
    java.lang.String str) throws java.io.IOException;
```

Parameters

str

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeChar Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeChar(
    int val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeChars Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeChars(
    java.lang.String str) throws java.io.IOException;
```

Parameters

str

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeDouble Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeDouble(
    double val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeFloat Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeFloat(
    float val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeInt Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeInt(
    int val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeLong Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeLong(
    long val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeShort Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeShort(
    int val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutput.writeUTF Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeUTF(
    java.lang.String str) throws java.io.IOException;
```

Parameters

str

See Also

Reference

DataOutput Interface

Concepts

DataOutput Members

java.io Package

DataOutputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.DataOutputStream
    extends java.io.FilterOutputStream
    implements java.io.DataOutput
```

Inheritance Hierarchy

java.lang.Object

  java.io.OutputStream

    java.io.FilterOutputStream

      java.io.DataOutputStream

See Also

Concepts

DataOutputStream Members

java.io Package

DataOutputStream Members

The following tables list the members exposed by the DataOutputStream type.

Public Constructors

Name	Description
DataOutputStream	

Public Fields

Name	Description
out	(inherited from FilterOutputStream)
written	

Public Methods

Name	Description
close	(inherited from FilterOutputStream)
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
size	
toString	(inherited from OutputStream)
write	Overloaded. Overridden.
writeBoolean	
writeByte	
writeBytes	
writeChar	
writeChars	
writeDouble	
writeFloat	
writeInt	

writeLong	
writeShort	
writeUTF	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DataOutputStream Class

Concepts

java.io Package

DataOutputStream Fields

Public Fields

Name	Description
out	(inherited from FilterOutputStream)
written	

See Also

Reference

DataOutputStream Class

Concepts

java.io Package

DataOutputStream.written Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int written;
```

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.DataOutputStream(
    java.io.OutputStream out);
```

Parameters

out

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream Methods

Public Methods

Name	Description
close	(inherited from FilterOutputStream)
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
size	
toString	(inherited from OutputStream)
write	Overloaded. Overridden.
writeBoolean	
writeByte	
writeBytes	
writeChar	
writeChars	
writeDouble	
writeFloat	
writeInt	
writeLong	
writeShort	
writeUTF	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DataOutputStream Class

Concepts
java.io Package

DataOutputStream.flush Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush() throws java.io.IOException;
```

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.size Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int size();
```

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.write Method

Overload List

Name	Description
DataOutputStream.write (int)	
DataOutputStream.write (byte[])	
DataOutputStream.write (byte[], int, int)	

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void write(
    int b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.write Method (SByte[], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void write(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeBoolean Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeBoolean(
    boolean v) throws java.io.IOException;
```

Parameters

v

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeByte Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeByte(
    int v) throws java.io.IOException;
```

Parameters

v

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeBytes Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeBytes(
    java.lang.String s) throws java.io.IOException;
```

Parameters

s

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeChar Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeChar(
    int v) throws java.io.IOException;
```

Parameters

v

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeChars Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeChars(
    java.lang.String s) throws java.io.IOException;
```

Parameters

s

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeDouble Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeDouble(
    double v) throws java.io.IOException;
```

Parameters

v

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeFloat Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeFloat(
    float v) throws java.io.IOException;
```

Parameters

v

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeInt Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeInt(
    int v) throws java.io.IOException;
```

Parameters

v

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeLong Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeLong(
    long v) throws java.io.IOException;
```

Parameters

v

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeShort Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeShort(
    int v) throws java.io.IOException;
```

Parameters

v

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

DataOutputStream.writeUTF Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeUTF(
    java.lang.String s) throws java.io.IOException;
```

Parameters

s

See Also

Reference

DataOutputStream Class

Concepts

DataOutputStream Members

java.io Package

EOFException Class

The exception that is thrown when an attempt is made to read past the end of a file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.EOFException
    extends java.io.IOException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.EOFException

See Also

Concepts

EOFException Members

java.io Package

EOFException Members

The exception that is thrown when an attempt is made to read past the end of a file.

The following tables list the members exposed by the EOFException type.

Public Constructors

Name	Description
EOFException	Overloaded. Initializes a new instance of an EOFException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

EOFException Class

Concepts

java.io Package

EOFException Constructor

Initializes a new instance of an EOFException object.

Overload List

Name	Description
EOFException ()	Initializes a new instance of an EOFException object.
EOFException (String)	Initializes a new instance of an EOFException object with the given message.
EOFException (SerializationInfo, StreamingContext)	Initializes a new instance of an EOFException object during deserialization.
EOFException (String, Exception)	Initializes a new instance of an EOFException object with the given message and inner exception.

See Also

Reference

EOFException Class

Concepts

EOFException Members

java.io Package

EOFException Constructor ()

Initializes a new instance of an EOFException object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.EOFException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

EOFException Class

Concepts

EOFException Members

java.io Package

EOFException Constructor (String)

Initializes a new instance of an EOFException object with the given message.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.EOFException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
EOFException Class
Concepts
EOFException Members
java.io Package

EOFException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an EOFException object during deserialization.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.EOFException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
EOFException Class
Concepts
EOFException Members
java.io Package

EOFException Constructor (String, Exception)

Initializes a new instance of an EOFException object with the given message and inner exception.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.EOFException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the EOFException.

See Also

Reference

EOFException Class

Concepts

EOFException Members

java.io Package

EOFException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

EOFException Class

Concepts

java.io Package

EOFException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

EOFException Class

Concepts

java.io Package

File Class

Contains methods to access, create, or edit files and folders on disk.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.File
    extends java.lang.Object
    implements java.io.Serializable, java.lang.Comparable, System.Runtime.Serialization.ISe
rializable
```

Example

The following example demonstrates the createNewFile, deleteOnExit, exists, getAbsolutePath, getCanonicalPath, getName, getParent, getPath, isDirectory, isFile, isHidden, lastModified, length, listRoots, renameTo, and toURL methods of the File class.

```
// file_overview.jsl

import java.io.*;
import java.net.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a file object using C:\AnyFile.txt.
            File anyFile = new File("C:\\Windows\\..\\AnyFile.txt");

            // If the file doesn't exist, create it now.
            if (!anyFile.exists())
            {
                if (!anyFile.createNewFile())
                {
                    System.out.println("Unable to create file " +
                        anyFile.getAbsolutePath() + "!");
                }
            }

            // Delete the file when we no longer need it.
            anyFile.deleteOnExit();

            // Display some useful information about this file.
            System.out.println(
                "getAbsolutePath: " + anyFile.getAbsolutePath() + "\n" +
                "getCanonicalPath: " + anyFile.getCanonicalPath() + "\n" +
                "getName: " + anyFile.getName() + "\n" +
                "getParent: " + anyFile.getParent() + "\n" +
                "getPath: " + anyFile.getPath() + "\n" +
                "isDirectory: " + anyFile.isDirectory() + "\n" +
                "isFile: " + anyFile.isFile() + "\n" +
                "isHidden: " + anyFile.isHidden() + "\n" +
                "lastModified: " + new Date(
                    anyFile.lastModified()).toString() + "\n" +
                "length: " + anyFile.length() + "\n" +
                "toURL: " + anyFile.toURL().toString());

            // List the root directories of the system.
            File[] roots = anyFile.listRoots();
            System.out.print("listRoots: ");
```

```
            for (int i = 0; i < roots.length; i++)
            {
                System.out.print(roots[i].getAbsolutePath() + " ");
            }

            // Rename the file.
            File newName = new File("C:\\NewName.txt");
            if (!anyFile.renameTo(newName))
            {
                System.out.println("\nError renaming " +
                    anyFile.getAbsolutePath() + " to " +
                    newName.getAbsolutePath());
            }
            else
            {
                System.out.println("\nrenameTo: " +
                    newName.getAbsolutePath());
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
getAbsolutePath: C:\Windows\..\AnyFile.txt
getCanonicalPath: C:\AnyFile.txt
getName: AnyFile.txt
getParent: C:\Windows\..
getPath: C:\Windows\..\AnyFile.txt
isDirectory: false
isFile: true
isHidden: false
lastModified: Fri Sep 17 13:14:37 PDT 2004
length: 0
toURL: file:/C:/AnyFile.txt
listRoots: A:\ C:\ D:\ J:\
renameTo: C:\NewName.txt
*/
```

Inheritance Hierarchy

java.lang.Object

java.io.File

See Also

Concepts

File Members

java.io Package

File Members

Contains methods to access, create, or edit files and folders on disk.

The following tables list the members exposed by the File type.

Public Constructors

Name	Description
File	Overloaded. Initializes a new instance of a File object.

Public Fields

Name	Description
pathSeparator	Represents the string used to separate elements in the PATH environment variable. On Windows, this is represented by a semicolon (;). On Unix, it is represented by a colon (:).
pathSeparatorChar	Represents the character used to separate elements in the PATH environment variable. On Windows, this is represented by a semicolon (;). On Unix, it is represented by a colon (:).
separator	Represents the string used to separate directories. On Windows, this is represented by a backslash (\). On Unix, it is represented by a forward slash (/).
separatorChar	Represents the character used to separate directories. On Windows, this is represented by a backslash (\). On Unix, it is represented by a forward slash (/).

Public Methods

Name	Description
canRead	Determines whether a file or folder can be read.
canWrite	Determines whether a file or folder can be written to.
compareTo	Overloaded. Compares two File objects for equality.
createNewFile	Creates on disk the file or folder that is represented by the File object.
createTempFile	Overloaded. Creates on disk a temporary file or folder.
delete	Deletes the file or folder represented by the File object.
deleteOnExit	Deletes the file or folder when the object is no longer needed and the system reclaims the object through garbage collection.
equals	Overridden. Compares two File objects for equality.
exists	Determines whether the file or folder exists on disk.
finalize	Overridden. Reclaims the memory associated with a File object when it is no longer needed by the application.
getAbsoluteFile	Creates a new instance of a File object representing the file located at the absolute path of the current File object.
getAbsolutePath	Displays the absolute path of the file.

getCanonicalFile	Creates a new instance of a File object representing the file located at the absolute path of the current File object. All '.' and '..' references will be resolved.
getCanonicalPath	Displays the absolute path of the file. All '.' and '..' references will be resolved.
hashCode	Overridden. Returns a hash value that is suitable for use in hashing algorithms.
getName	Retrieves the name of the file without the directory.
GetObjectData	Retrieves the data to be serialized.
getParent	Retrieves the path name of the parent directory of the file.
getParentFile	Creates a new instance of a File object representing the parent directory of the file.
getPath	Displays the directory of the file.
getClass	(inherited from Object)
isAbsolute	Determines whether the pathname of the file or folder is absolute.
isDirectory	Determines whether the pathname of the object represented by a File object is a directory.
isFile	Determines whether the pathname of the object represented by a File object is a file.
isHidden	Determines whether the file or folder is hidden.
lastModified	Retrieves the time of the last modification to the file or folder, in milliseconds since epoch.
length	Retrieves the size of the file or folder, in bytes.
list	Overloaded. Lists the files or folders in the directory represented by this object.
listFiles	Overloaded. Creates instances of new File objects representing all the files or folders in the directory represented by this object.
listRoots	Creates instances of new File objects representing all the base, or root, paths of the system.
clone	Creates a shallow copy of the current File object.
mkdir	Creates the directory on disk represented by this object.
mkdirs	Creates the directory and all parent directories on disk represented by this object.
renameTo	Renames the file or folder to the name of the destination file or folder.
setLastModified	Sets the time of the last modification to the file or folder.
setReadOnly	Sets the value indicating whether the file or folder is read-only.
toString	Overridden. Displays a human readable representation of a file or folder.
toURL	Converts a file or folder into a URL (Uniform Resource Locator).

See Also

Reference

File Class

Concepts

java.io Package

File Fields

Public Fields

Name	Description
pathSeparator	Represents the string used to separate elements in the PATH environment variable. On Windows, this is represented by a semicolon (;). On Unix, it is represented by a colon (:).
pathSeparatorChar	Represents the character used to separate elements in the PATH environment variable. On Windows, this is represented by a semicolon (;). On Unix, it is represented by a colon (:).
separator	Represents the string used to separate directories. On Windows, this is represented by a backslash (\). On Unix, it is represented by a forward slash (/).
separatorChar	Represents the character used to separate directories. On Windows, this is represented by a backslash (\). On Unix, it is represented by a forward slash (/).

See Also

Reference

File Class

Concepts

java.io Package

File.pathSeparator Field

Represents the string used to separate elements in the PATH environment variable. On Windows, this is represented by a semicolon (;). On Unix, it is represented by a colon (:).

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.String pathSeparator;
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.pathSeparatorChar Field

Represents the character used to separate elements in the PATH environment variable. On Windows, this is represented by a semicolon (;). On Unix, it is represented by a colon (:).

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final char pathSeparatorChar;
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.separator Field

Represents the string used to separate directories. On Windows, this is represented by a backslash (\). On Unix, it is represented by a forward slash (/).

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.String separator;
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.separatorChar Field

Represents the character used to separate directories. On Windows, this is represented by a backslash (\). On Unix, it is represented by a forward slash (/).

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final char separatorChar;
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File Constructor

Initializes a new instance of a File object.

Overload List

Name	Description
File (String)	Initializes a new instance of a File object at the given path.
File (File, String)	Initializes a new instance of a File object at the given directory and with the given name.
File (SerializationInfo, StreamingContext)	Initializes a new instance of a File object during deserialization.
File (String, String)	Initializes a new instance of a File object at the given directory and with the given name.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File Constructor (String)

Initializes a new instance of a File object at the given path.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.File(
    java.lang.String path);
```

Parameters

path

The name of the file or folder to be accessed or created. This can be an absolute or relative value.

See Also
Reference
File Class
Concepts
File Members
java.io Package

File Constructor (File, String)

Initializes a new instance of a File object at the given directory and with the given name.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.File(
    java.io.File dir,
    java.lang.String name);
```

Parameters

dir

A File object containing the directory of the file or folder.

name

The name of the file or folder to be accessed or created in the given directory.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a File object during deserialization.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.File(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File Constructor (String, String)

Initializes a new instance of a File object at the given directory and with the given name.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.File(
    java.lang.String dirPath,
    java.lang.String name);
```

Parameters

dirPath

The directory containing the file or folder.

name

The name of the file or folder to be accessed or created in the given directory.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File Methods

Public Methods

Name	Description
canRead	Determines whether a file or folder can be read.
canWrite	Determines whether a file or folder can be written to.
compareTo	Overloaded. Compares two File objects for equality.
createNewFile	Creates on disk the file or folder that is represented by the File object.
createTempFile	Overloaded. Creates on disk a temporary file or folder.
delete	Deletes the file or folder represented by the File object.
deleteOnExit	Deletes the file or folder when the object is no longer needed and the system reclaims the object through garbage collection.
equals	Overridden. Compares two File objects for equality.
exists	Determines whether the file or folder exists on disk.
finalize	Overridden. Reclaims the memory associated with a File object when it is no longer needed by the application.
getAbsoluteFile	Creates a new instance of a File object representing the file located at the absolute path of the current File object.
getAbsolutePath	Displays the absolute path of the file.
getCanonicalFile	Creates a new instance of a File object representing the file located at the absolute path of the current File object. All '.' and '..' references will be resolved.
getCanonicalPath	Displays the absolute path of the file. All '.' and '..' references will be resolved.
hashCode	Overridden. Returns a hash value that is suitable for use in hashing algorithms.
getName	Retrieves the name of the file without the directory.
GetObjectData	Retrieves the data to be serialized.
getParent	Retrieves the path name of the parent directory of the file.
getParentFile	Creates a new instance of a File object representing the parent directory of the file.
getPath	Displays the directory of the file.
getClass	(inherited from Object)
isAbsolute	Determines whether the pathname of the file or folder is absolute.

isDirectory	Determines whether the pathname of the object represented by a File object is a directory.
isFile	Determines whether the pathname of the object represented by a File object is a file.
isHidden	Determines whether the file or folder is hidden.
lastModified	Retrieves the time of the last modification to the file or folder, in milliseconds since epoch.
length	Retrieves the size of the file or folder, in bytes.
list	Overloaded. Lists the files or folders in the directory represented by this object.
listFiles	Overloaded. Creates instances of new File objects representing all the files or folders in the directory represented by this object.
listRoots	Creates instances of new File objects representing all the base, or root, paths of the system.
clone	Creates a shallow copy of the current File object.
mkdir	Creates the directory on disk represented by this object.
mkdirs	Creates the directory and all parent directories on disk represented by this object.
renameTo	Renames the file or folder to the name of the destination file or folder.
setLastModified	Sets the time of the last modification to the file or folder.
setReadOnly	Sets the value indicating whether the file or folder is read-only.
toString	Overridden. Displays a human readable representation of a file or folder.
toURL	Converts a file or folder into a URL (Uniform Resource Locator).

See Also

Reference

File Class

Concepts

java.io Package

File.canRead Method

Determines whether a file or folder can be read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean canRead();
```

Return Value

true if the file or folder can be read; false otherwise.

Example

The following code example demonstrates how to determine if a file is readable.

```
// file_canread.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt as read-write.
        File myFile = new File("C:\\MyFile.txt");

        // Determine if this file is readable.
        if (myFile.canRead())
        {
            System.out.println(myFile.getAbsolutePath() +
                " is a readable file.");
        }
    }
}

/*
Output:
C:\MyFile.txt is a readable file.
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.canWrite Method

Determines whether a file or folder can be written to.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean canWrite();
```

Return Value

true if the file or folder can be written to; false otherwise.

Example

The following example demonstrates how to determine if a file has read-write access.

```
// file_canwrite.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt as read-write.
        File myFile = new File("C:\\MyFile.txt");

        // If this file is read-write, set it to read-only.
        if (myFile.canWrite())
        {
            myFile.setReadOnly();
            System.out.println(myFile.getAbsolutePath() +
                " is now read-only.");
        }
    }
}

/*
Output:
C:\MyFile.txt is now read-only.
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.clone Method

Creates a shallow copy of the current File object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

Return Value

A shallow copy of the current File object.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.compareTo Method

Compares two File objects for equality.

Overload List

Name	Description
File.compareTo (File)	Compares one File object with another File object.
File.compareTo (Object)	Compares one File object with another object.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.compareTo Method (File)

Compares one File object with another File object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.io.File pathname);
```

Parameters

pathname

A File object representing a file or folder to compare against.

Return Value

Less than zero if the path and name of the object calling compareTo is alphabetically less than pathname, 0 if the path and name of the object calling compareTo is alphabetically equal to pathname, and greater than zero if the path and name of the object calling compareTo is alphabetically greater than pathname.

Example

The following example demonstrates how to compare two file names alphabetically.

```
// file_compareto.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Open the file C:\YourFile.txt.
        File yourFile = new File("C:\\YourFile.txt");

        if (myFile.exists() && yourFile.exists())
        {
            // Compare the two files alphabetically.
            int compareResult = myFile.compareTo(yourFile);

            // Display the results of the comparison.
            if (compareResult < 0)
            {
                System.out.println(myFile.getAbsolutePath() +
                    " is less than " +
                    yourFile.getAbsolutePath());
            }
            else if (compareResult == 0)
            {
                System.out.println(myFile.getAbsolutePath() +
                    " is equal to " +
                    yourFile.getAbsolutePath());
            }
            else
            {
                System.out.println(myFile.getAbsolutePath() +
                    " is greater than " +
                    yourFile.getAbsolutePath());
            }
        }
```

```
    }
}

/*
Output:
C:\MyFile.txt is less than C:\YourFile.txt
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.compareTo Method (Object)

Compares one File object with another object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object o);
```

Parameters

o

An object representing a file or folder to compare against.

Return Value

Less than zero if the path and name of the object calling compareTo is alphabetically less than the object o, 0 if the path and name of the object calling compareTo is alphabetically equal to the object o, and greater than zero if the path and name of the object calling compareTo is alphabetically greater than the object o.

Example

The following example demonstrates how to compare two file names alphabetically.

```
// file_compareto_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt.
        Object myFile = new File("C:\\MyFile.txt");

        // Open the file C:\YourFile.txt.
        Object yourFile = new File("C:\\YourFile.txt");

        if (myFile instanceof File &&
            yourFile instanceof File &&
            ((File)myFile).exists() && ((File)yourFile).exists())
        {
            // Compare the two files alphabetically.
            int compareResult = ((File)myFile).compareTo(yourFile);

            // Display the results of the comparison.
            if (compareResult < 0)
            {
                System.out.println(((File)myFile).getAbsolutePath() +
                    " is less than " +
                    ((File)yourFile).getAbsolutePath());
            }
            else if (compareResult == 0)
            {
                System.out.println(((File)myFile).getAbsolutePath() +
                    " is equal to " +
                    ((File)yourFile).getAbsolutePath());
            }
            else
            {
                System.out.println(((File)myFile).getAbsolutePath() +
                    " is greater than " +
                    ((File)yourFile).getAbsolutePath());
```

```
            }
        }
    }
}

/*
Output:
C:\MyFile.txt is less than C:\YourFile.txt
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.createNewFile Method

Creates on disk the file or folder that is represented by the File object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean createNewFile() throws java.io.IOException;
```

Return Value

true if the file or folder was successfully created; false if the file or folder already exists or there was an error creating it.

Example

The following example shows how to create a file that does not yet exist.

```
// file_createnewfile.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Open the file C:\MyFile.txt.
            File myFile = new File("C:\\MyFile.txt");
            if (myFile.exists())
            {
                System.out.println(myFile.getAbsolutePath() +
                    " already exists.");
            }
            else
            {
                if (myFile.createNewFile())
                {
                    System.out.println("Created file " +
                        myFile.getAbsolutePath());
                }
                else
                {
                    System.out.println("Unable to create file " +
                        myFile.getAbsolutePath());
                }
            }

            // Open the file C:\DoesNotExist.txt.
            File noFile = new File ("C:\\DoesNotExist.txt");
            if (noFile.exists())
            {
                System.out.println(noFile.getAbsolutePath() +
                    " already exists.");
            }
            else
            {
                if (noFile.createNewFile())
                {
                    System.out.println("Created file " +
                        noFile.getAbsolutePath());
                }
                else
                {
```

```
                    System.out.println("Unable to create file " +
                        noFile.getAbsolutePath());
                }
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
C:\MyFile.txt already exists.
Created file C:\DoesNotExist.txt
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.createTempFile Method

Creates on disk a temporary file or folder.

Overload List

Name	Description
File.createTempFile (String, String)	Creates on disk a temporary file or folder with the given prefix (name) and suffix (extension).
File.createTempFile (String, String, File)	Creates on disk a temporary file with the given prefix (name) and suffix (extension) located in the given directory.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.createTempFile Method (String, String)

Creates on disk a temporary file or folder with the given prefix (name) and suffix (extension).

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static java.io.File createTempFile(
    java.lang.String prefix,
    java.lang.String suffix) throws java.io.IOException;
```

Parameters

prefix

The name of the temporary file to create. This value will be modified to add a trailing string of random characters when it is written to disk.

suffix

The extension, or type, of the file to create. This suffix should include the leading period or it will not be added.

Return Value

A File object representing the newly created temporary file. The temporary file is created in the directory represented by GetTempPath.

Example

The following example shows how to create a temporary file.

```
// file_createtempfile.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a temp file named ATempFile.tmp.
            File tempFile = File.createTempFile("ATempFile", ".tmp");

            // Determine where this file was created.
            if (tempFile.exists())
            {
                System.out.println("A temporary file was created at " +
                    tempFile.getAbsolutePath());
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
A temporary file was created at C:\Documents and Settings\yourname\Local Settings\Temp\ATem
pFiletmp3
2B244356857.tmp
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.createTempFile Method (String, String, File)

Creates on disk a temporary file with the given prefix (name) and suffix (extension) located in the given directory.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static java.io.File createTempFile(
    java.lang.String prefix,
    java.lang.String suffix,
    java.io.File directory) throws java.io.IOException;
```

Parameters

prefix

The name of the temporary file to create. This value will be modified to add a trailing string of random characters when it is written to disk.

suffix

The extension, or type, of the file to create. This suffix should include the leading period or it will not be added.

directory

The directory where the temporary file is to be created.

Return Value

A File object representing the newly created temporary file.

Example

The following example shows how to create a temporary file in the C:\ directory.

```
// file_createtempfile_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a temp file named ATempFile.tmp.
            File dir = new File("C:\\");
            if (dir.exists())
            {
                File tempFile = File.createTempFile("ATempFile", ".tmp", dir);

                // Determine where this file was created.
                if (tempFile.exists())
                {
                    System.out.println("A temporary file was created at " +
                        tempFile.getAbsolutePath());
                }
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
A temporary file was created at C:\ATempFiletmp32E1187765574.tmp
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.delete Method

Deletes the file or folder represented by the File object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean delete();
```

Return Value

true if the file or folder was successfully deleted; false if the file or folder does not exist or could not be deleted.

Example

The following example shows how to delete a file.

```
// file_delete.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");
        String path = myFile.getAbsolutePath();

        // If the file exists, delete it.
        if (myFile.exists())
        {
            if (myFile.delete())
            {
                System.out.println("Successfully deleted " +
                    path);
            }
            else
            {
                System.out.println("Unable to delete file " +
                    path);
            }
        }
    }
}

/*
Output:
Successfully deleted C:\MyFile.txt
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.deleteOnExit Method

Deletes the file or folder when the object is no longer needed and the system reclaims the object through garbage collection.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void deleteOnExit();
```

Example

The following example shows how to delete a file once the system is no longer using it.

```
// file_deleteonexit.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");
        String path = myFile.getAbsolutePath();

        // If the file exists, delete it when garbage collection
        // runs on the myFile object.
        if (myFile.exists())
        {
            myFile.deleteOnExit();
        }
    }
}
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.equals Method

Compares two File objects for equality.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

An object representing a file or folder to compare against.

Return Value

true if the two File objects represent the same file or folder; false otherwise.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.exists Method

Determines whether the file or folder exists on disk.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean exists();
```

Return Value

true if the file or folder represented by the File object exists on disk; false otherwise.

Example

The following example shows how to determine if a file exists.

```
// file_exists.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");
        String path = myFile.getAbsolutePath();

        // If the file exists, display it's full path.
        if (myFile.exists())
        {
            System.out.println(path + " exists!");
        }
        else
        {
            System.out.println(path + " does NOT exist!");
        }
    }
}

/*
Output:
C:\MyFile.txt exists!
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.finalize Method

Reclaims the memory associated with a File object when it is no longer needed by the application.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void finalize();
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.getAbsoluteFile Method

Creates a new instance of a File object representing the file located at the absolute path of the current File object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.File getAbsoluteFile();
```

Return Value

A new instance of a File object representing the file located at the absolute path of the current File object.

Example

The following example shows how to create a new File object by calling getAbsoluteFile.

```
// file_getabsolutefile.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");
        File myFile2 = myFile.getAbsoluteFile();
        String path = myFile2.getAbsolutePath();

        // If the file exists, display it's full path.
        if (myFile2.exists())
        {
            System.out.println(path + " exists!");
        }
        else
        {
            System.out.println(path + " does not exist!");
        }
    }
}

/*
Output:
C:\MyFile.txt exists!
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.getAbsolutePath Method

Displays the absolute path of the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getAbsolutePath();
```

Return Value

The absolute path of the file represented by the File object.

Example

The following example displays the absolute path for a file.

```
// file_getabsolutepath.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt.
        File myFile = new File("C:\\Windows\\..\\MyFile.txt");
        String path = myFile.getAbsolutePath();

        // If the file exists, display it's full path.
        if (myFile.exists())
        {
            System.out.println(path + " exists!");
        }
        else
        {
            System.out.println(path + " does NOT exist!");
        }
    }
}

/*
Output:
C:\Windows\..\MyFile.txt exists!
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.getCanonicalFile Method

Creates a new instance of a File object representing the file located at the absolute path of the current File object. All '.' and '..' references will be resolved.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.File getCanonicalFile() throws java.io.IOException;
```

Return Value

A new instance of a File object representing the file located at the absolute path of the current File object.

Example

The following example shows how to create a new File object by calling getCanonicalFile.

```
// file_getcanonicalfile.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Open the file C:\MyFile.txt.
            File myFile = new File("C:\\MyFile.txt");
            File myFile2 = myFile.getCanonicalFile();
            String path = myFile2.getAbsolutePath();

            // If the file exists, display it's full path.
            if (myFile2.exists())
            {
                System.out.println(path + " exists!");
            }
            else
            {
                System.out.println(path + " does not exist!");
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
C:\MyFile.txt exists!
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.getCanonicalPath Method

Displays the absolute path of the file. All '.' and '..' references will be resolved.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getCanonicalPath() throws java.io.IOException;
```

Return Value

The absolute path of the file represented by the File object.

Example

The following example displays the canonical path for a file.

```
// file_getcanonicalpath.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Open the file C:\MyFile.txt.
            File myFile = new File("C:\\Windows\\..\\MyFile.txt");
            String path = myFile.getCanonicalPath();

            // If the file exists, display it's full path.
            if (myFile.exists())
            {
                System.out.println(path + " exists!");
            }
            else
            {
                System.out.println(path + " does NOT exist!");
            }
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
C:\MyFile.txt exists!
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.hashCode Method

Returns a hash value that is suitable for use in hashing algorithms.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

A hash value that is suitable for use in hashing algorithms.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.getName Method

Retrieves the name of the file without the directory.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getName();
```

Return Value

The name of the file represented by the File object without the directory.

Example

The following example displays the name of a file.

```
// file_getname.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Display the name of the file.
        System.out.println("The file's name is " +
            myFile.getName());
    }
}

/*
Output:
The file's name is MyFile.txt
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.GetObjectData Method

Retrieves the data to be serialized.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization process.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.getParent Method

Retrieves the path name of the parent directory of the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getParent();
```

Return Value

The path name of the parent directory of the file represented by the File object.

Example

The following example displays the parent of a file.

```
// file_getparent.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Display the name of the file and the parent.
        System.out.println("The file's name is " +
            myFile.getName() + ".\nThe parent's name is " +
            myFile.getParent());
    }
}

/*
Output:
The file's name is MyFile.txt.
The parent's name is C:\
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.getParentFile Method

Creates a new instance of a File object representing the parent directory of the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.File getParentFile();
```

Return Value

A new instance of a File object representing the parent directory of the file represented by the File object, or null if the parent directory does not exist.

Example

The following example demonstrates how to create a new File object representing the parent of a given File object.

```
// file_getparentfile.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");
        File parentFile = myFile.getParentFile();

        // Display the name of the file and the parent.
        if (myFile.exists() &&
            (parentFile != null) &&
            parentFile.exists())
        {
            System.out.println("The file's name is " +
                myFile.getName() + ".\nThe parent's path is " +
                parentFile.getAbsolutePath());
        }
    }
}

/*
Output:
The file's name is MyFile.txt.
The parent's path is C:\
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.getPath Method

Displays the directory of the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getPath();
```

Return Value

The directory of the file represented by the File object.

Example

The following example shows how to display the path for a file.

```
// file_getpath.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt.
        File myFile = new File("C:\\Windows\\..\\MyFile.txt");
        String path = myFile.getPath();

        // If the file exists, display it's full path.
        if (myFile.exists())
        {
            System.out.println(path + " exists!");
        }
        else
        {
            System.out.println(path + " does NOT exist!");
        }
    }
}

/*
Output:
C:\Windows\..\MyFile.txt exists!
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.isAbsolute Method

Determines whether the pathname of the file or folder is absolute.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean isAbsolute();
```

Return Value

true if the pathname of the file or folder represented by the File object is absolute; false otherwise.

Example

The following example shows how to determine if a file path is absolute.

```
// file_isabsolute.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Create a File using YourFile.txt.
        File yourFile = new File("YourFile.txt");

        // Which of these files have absolute paths?
        if (myFile.isAbsolute())
        {
            System.out.println(myFile.getName() +
                " is absolute.");
        }
        else
        {
            System.out.println(myFile.getName() +
                " is NOT absolute.");
        }

        if (yourFile.isAbsolute())
        {
            System.out.println(yourFile.getName() +
                " is absolute.");
        }
        else
        {
            System.out.println(yourFile.getName() +
                " is NOT absolute.");
        }
    }
}

/*
Output:
MyFile.txt is absolute.
YourFile.txt is NOT absolute.
*/
```

See Also

Reference
File Class
Concepts
File Members
java.io Package

File.isDirectory Method

Determines whether the pathname of the object represented by a File object is a directory.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean isDirectory();
```

Return Value

true if the object represented by the File object exists and is a directory; false otherwise.

Example

The following example demonstrates how to determine if a File object is representing a directory (folder).

```
// file_isdirectory.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Get the parent of MyFile.txt.
        File parentFile = myFile.getParentFile();

        // Which of these is a file and which is a directory?
        if (myFile.isDirectory())
        {
            System.out.println(myFile.getAbsolutePath() +
                " is a directory.");
        }
        else
        {
            System.out.println(myFile.getAbsolutePath() +
                " is NOT a directory.");
        }

        if (parentFile.isDirectory())
        {
            System.out.println(parentFile.getAbsolutePath() +
                " is a directory.");
        }
        else
        {
            System.out.println(parentFile.getAbsolutePath() +
                " is NOT a directory.");
        }
    }
}

/*
Output:
C:\MyFile.txt is NOT a directory.
C:\ is a directory.
*/
```

See Also

Reference
File Class
Concepts
File Members
java.io Package

File.isFile Method

Determines whether the pathname of the object represented by a File object is a file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean isFile();
```

Return Value

true if the object represented by the File object exists and is a file; false otherwise.

Example

The following example demonstrates how to determine if a File object is representing a file (rather than a directory).

```
// file_isfile.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Get the parent of MyFile.txt.
        File parentFile = myFile.getParentFile();

        // Which of these is a file and which is a directory?
        if (myFile.isFile())
        {
            System.out.println(myFile.getAbsolutePath() +
                " is a file.");
        }
        else
        {
            System.out.println(myFile.getAbsolutePath() +
                " is NOT a file.");
        }

        if (parentFile.isFile())
        {
            System.out.println(parentFile.getAbsolutePath() +
                " is a file.");
        }
        else
        {
            System.out.println(parentFile.getAbsolutePath() +
                " is NOT a file.");
        }
    }
}

/*
Output:
C:\MyFile.txt is a file.
C:\ is NOT a file.
*/
```

See Also

Reference
File Class
Concepts
File Members
java.io Package

File.isHidden Method

Determines whether the file or folder is hidden.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean isHidden();
```

Return Value

true if the file or folder represented by the File object is hidden; false otherwise.

Example

The following example demonstrates how to determine if a file is hidden.

```
// file_ishidden.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Determine if the file is hidden.
        if (myFile.isHidden())
        {
            System.out.println(myFile.getAbsolutePath() +
                " is a hidden file.");
        }
        else
        {
            System.out.println(myFile.getAbsolutePath() +
                " is NOT a hidden file.");
        }
    }
}

/*
Output:
C:\MyFile.txt is NOT a hidden file.
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.lastModified Method

Retrieves the time of the last modification to the file or folder, in milliseconds since epoch.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long lastModified();
```

Return Value

The time of the last modification to the file represented by the File object, in milliseconds since epoch, or -1 if the file or folder does not exist.

Example

The following example displays the time of the last modification on a file.

```
// file_lastmodified.jsl

import java.io.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Determine the last modification time of the file.
        long modTime = myFile.lastModified();
        Date d = new Date(modTime);

        System.out.println(myFile.getAbsolutePath() +
            " was last modified on " + d.toString());
    }
}

/*
Output:
C:\MyFile.txt was last modified on Fri Sep 17 10:51:45 PDT 2004
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.length Method

Retrieves the size of the file or folder, in bytes.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long length();
```

Return Value

The size of the file or folder represented by the File object, in bytes, or 0 if the file or folder does not exist.

Example

The following example displays the length of a file in bytes.

```
// file_length.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Determine the length of the file.
        long length = myFile.length();

        System.out.println(myFile.getAbsolutePath() +
            " contains " + length + " bytes.");
    }
}

/*
Output:
C:\MyFile.txt contains 66 bytes.
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.list Method

Lists the files or folders in the directory represented by this object.

Overload List

Name	Description
File.list ()	Lists the files or folders in the directory represented by this object.
File.list (FilenameFilter)	Lists the files or folders in the directory represented by this object that meet the filtering criteria.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.list Method ()

Lists the files or folders in the directory represented by this object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String[] list();
```

Return Value

An array of all the files or folders in the directory represented by this object, or null if the object does not exist or is not a directory.

Example

The following example displays the contents of the C:\ directory.

```
// file_list.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Determine all the other folders and files in the
        // same directory as myFile.
        String[] contents = myFile.getParentFile().list();

        System.out.println("The parent directory of " +
            myFile.getPath() +
            " contains files and folders:");
        for (int i = 0; i < contents.length; i++)
        {
            System.out.println("\t" + contents[i]);
        }
    }
}

/*
Output:
The parent directory of C:\MyFile.txt contains files and folders:
        dd
        dell
        Documents and Settings
        Drivers
        Inetpub
        MSOCache
        Perl
        Program Files
        Python23
        RECYCLER
        System Volume Information
        WINNT
        wmpub
        WUTemp
        AUTOEXEC.BAT
        boot.ini
        CONFIG.SYS
```

```
          DoesNotExist.txt
          InoSetRTThread.log
          IO.SYS
          MSDOS.SYS
          MyFile.txt
          NTDETECT.COM
          ntldr
          pagefile.sys
          YourFile.txt
  */
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.list Method (FilenameFilter)

Lists the files or folders in the directory represented by this object that meet the filtering criteria.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String[] list(
    java.io.FilenameFilter filter);
```

Parameters

filter

A FilenameFilter that represents the filter to apply to all the files or folders in the directory represented by this object.

Return Value

An array of all the files or folders in the directory represented by this object that meet the filtering criteria, or null if the object does not exist or is not a directory.

Example

The following example displays the files in the C:\ directory that have a txt extension.

```
// file_list_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Determine all the other folders and files in the
        // same directory as myFile.
        TxtFilenameFilter filter = new TxtFilenameFilter();
        String[] contents = myFile.getParentFile().list(filter);

        System.out.println("The parent directory of " +
            myFile.getPath() +
            " contains files and folders:");
        for (int i = 0; i < contents.length; i++)
        {
            System.out.println("\t" + contents[i]);
        }
    }
}

public class TxtFilenameFilter implements FilenameFilter
{
    public boolean accept(File dir, String name)
    {
        if (name != null &&
            name.toLowerCase().endsWith(".txt"))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

```
}

/*
Output:
The parent directory of C:\MyFile.txt contains files and folders:
        DoesNotExist.txt
        MyFile.txt
        NewName.txt
        output.txt
        RandomFile.txt
        YourFile.txt
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.listFiles Method

Creates instances of new File objects representing all the files or folders in the directory represented by this object.

Overload List

Name	Description
File.listFiles ()	Creates instances of new File objects representing all the files or folders in the directory represented by this object.
File.listFiles (FileFilter)	Creates instances of new File objects representing all the files or folders in the directory represented by this object that meet the file filtering criteria.
File.listFiles (FilenameFilter)	Creates instances of new File objects representing all the files or folders in the directory represented by this object that meet the filename filtering criteria.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.listFiles Method ()

Creates instances of new File objects representing all the files or folders in the directory represented by this object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.File[] listFiles();
```

Return Value

An array of File objects representing all the files or folders in the directory represented by this object, or null if the object does not exist or is not a directory.

Example

The following example displays the contents of the C:\ directory.

```
// file_listfiles.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Determine all the other folders and files in the
        // same directory as myFile.
        File[] contents = myFile.getParentFile().listFiles();

        System.out.println("The parent directory of " +
            myFile.getPath() +
            " contains files and folders:");
        for (int i = 0; i < contents.length; i++)
        {
            System.out.println("\t" + contents[i].getName());
        }
    }
}

/*
Output:
The parent directory of C:\MyFile.txt contains files and folders:
        dd
        dell
        Documents and Settings
        Drivers
        Inetpub
        MSOCache
        Perl
        Program Files
        Python23
        RECYCLER
        System Volume Information
        WINNT
        wmpub
        WUTemp
        AUTOEXEC.BAT
        boot.ini
        CONFIG.SYS
```

```
          DoesNotExist.txt
          InoSetRTThread.log
          IO.SYS
          MSDOS.SYS
          MyFile.txt
          NTDETECT.COM
          ntldr
          pagefile.sys
          YourFile.txt
  */
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.listFiles Method (FileFilter)

Creates instances of new File objects representing all the files or folders in the directory represented by this object that meet the file filtering criteria.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.File[] listFiles(
    java.io.FileFilter filter);
```

Parameters

filter

A FileFilter that represents the filter to apply to all the files or folders in the directory represented by this object.

Return Value

An array of File objects representing all the files or folders in the directory represented by this object that meet the filtering criteria, or null if the object does not exist or is not a directory.

Example

The following example displays the files in the C:\ directory that have a txt extension.

```
// file_listfiles_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Determine all the other folders and files in the
        // same directory as myFile.
        TxtFileFilter filter = new TxtFileFilter();
        File[] contents = myFile.getParentFile().listFiles(filter);

        System.out.println("The parent directory of " +
            myFile.getPath() +
            " contains files and folders:");
        for (int i = 0; i < contents.length; i++)
        {
            System.out.println("\t" + contents[i].getName());
        }
    }
}

public class TxtFileFilter implements FileFilter
{
    public boolean accept(File f)
    {
        if (f != null &&
            f.getName().toLowerCase().endsWith(".txt"))
        {
            return true;
        }
        else
        {
            return false;
        }
```

```
    }

    public String getDescription()
    {
        return "Filter for all txt files.";
    }
}

/*
Output:
The parent directory of C:\MyFile.txt contains files and folders:
        DoesNotExist.txt
        MyFile.txt
        NewName.txt
        output.txt
        RandomFile.txt
        YourFile.txt
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.listFiles Method (FilenameFilter)

Creates instances of new File objects representing all the files or folders in the directory represented by this object that meet the filename filtering criteria.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.File[] listFiles(
    java.io.FilenameFilter filter);
```

Parameters

filter

A FilenameFilter that represents the filter to apply to all the files or folders in the directory represented by this object.

Return Value

An array of File objects representing all the files or folders in the directory represented by this object that meet the filtering criteria, or null if the object does not exist or is not a directory.

Example

The following example displays the files in the C:\ directory that have a txt extension.

```
// file_listfiles_3.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Determine all the other folders and files in the
        // same directory as myFile.
        TxtFilenameFilter filter = new TxtFilenameFilter();
        File[] contents = myFile.getParentFile().listFiles(filter);

        System.out.println("The parent directory of " +
            myFile.getPath() +
            " contains files and folders:");
        for (int i = 0; i < contents.length; i++)
        {
            System.out.println("\t" + contents[i].getName());
        }
    }
}

public class TxtFilenameFilter implements FilenameFilter
{
    public boolean accept(File dir, String name)
    {
        if (name != null &&
            name.toLowerCase().endsWith(".txt"))
        {
            return true;
        }
        else
        {
            return false;
        }
```

```
    }
}

/*
Output:
The parent directory of C:\MyFile.txt contains files and folders:
        DoesNotExist.txt
        MyFile.txt
        NewName.txt
        output.txt
        RandomFile.txt
        YourFile.txt
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.listRoots Method

Creates instances of new File objects representing all the base, or root, paths of the system.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static java.io.File[] listRoots();
```

Return Value

An array of File objects representing all the base, or root, paths of the system. Paths that the user does not have permission to access are not included in the array.

Example

The following example lists all the root paths of the system.

```
// file_listroots.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Determine all the root paths for the system.
        File[] roots = File.listRoots();

        System.out.println("The system contains root paths:");
        for (int i = 0; i < roots.length; i++)
        {
            System.out.println("\t" + roots[i].getAbsolutePath());
        }
    }
}

/*
Output:
The system contains root paths:
        A:\
        C:\
        D:\
        J:\
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.mkdir Method

Creates the directory on disk represented by this object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean mkdir();
```

Return Value

true if the directory was successfully created, false otherwise.

Example

The following example shows how to create the parent directory of a file.

```
// file_mkdir.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\NewDirectory\MyFile.txt.
        File myFile = new File("C:\\NewDirectory", "MyFile.txt");

        // Create "New Directory" if it doesn't exist.
        if (myFile.mkdir())
        {
            System.out.println(myFile.getParent() +
                " was successfully created.");
        }
        else
        {
            System.out.println(myFile.getParent() +
                " was NOT created.");
        }
    }
}

/*
Output:
C:\NewDirectory was successfully created.
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.mkdirs Method

Creates the directory and all parent directories on disk represented by this object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean mkdirs();
```

Return Value

true if the directory was successfully created along with all parent directories, false otherwise.

Example

The following example shows how to create the parent directories of a file.

```
// file_mkdirs.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\NewDirectory\MyFile.txt.
        File myFile = new File("C:\\NewDirectory1\\NewDirectory2",
            "MyFile.txt");

        // Create "New Directory" if it doesn't exist.
        if (myFile.mkdir())
        {
            System.out.println(myFile.getParent() +
                " was successfully created.");
        }
        else
        {
            System.out.println(myFile.getParent() +
                " was NOT created.");
        }
    }
}

/*
Output:
C:\NewDirectory1\NewDirectory2 was successfully created.
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.renameTo Method

Renames the file or folder to the name of the destination file or folder.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean renameTo(
    java.io.File dest);
```

Parameters

dest

A File object representing the new name of the file or folder.

Return Value

true if the file or folder was successfully renamed; false otherwise.

Example

The following example shows how to rename a file.

```
// file_renameto.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Open the file C:\MyFile.txt.
        File myFile = new File("C:\\Windows\\..\\MyFile.txt");
        File newFile = new File("C:\\NewFile.txt");

        // If the file exists, rename it.
        if (myFile.exists())
        {
            myFile.renameTo(newFile);
            System.out.println("The new name is: " +
                newFile.getAbsolutePath());
        }
        else
        {
            System.out.println(myFile.getAbsolutePath() +
                " does NOT exist!");
        }
    }
}

/*
Output:
The new name is: C:\NewFile.txt
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.setLastModified Method

Sets the time of the last modification to the file or folder.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean setLastModified(
    long time);
```

Parameters

time

A value representing the time of the last modification to the file or folder, in milliseconds since epoch

Return Value

true if the modification time of the file or folder was successfully changed; false otherwise.

Example

The following example shows how to change the last modification time for a file.

```
// file_setlastmodified.jsl

import java.io.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Change the last modification time of the file to the
        // current time.
        Date d = new Date();
        System.out.println("Setting the date to " +
            d.toString());
        boolean changed = myFile.setLastModified(d.getTime());
    }
}

/*
Output:
Setting the date to Fri Sep 17 12:42:24 PDT 2004
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.setReadOnly Method

Sets the value indicating whether the file or folder is read-only.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean setReadOnly();
```

Return Value

true if the file or folder was successfully set to read-only; false if the file or folder does not exist.

Example

The following example shows how to make a file read-only.

```
// file_setreadonly.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a File object using C:\MyFile.txt.
        File myFile = new File("C:\\MyFile.txt");

        // Set the file to be read-only.
        boolean changed = myFile.setReadOnly();
        if (changed)
        {
            System.out.println(myFile.getAbsolutePath() +
                " is now read-only.");
        }
        else
        {
            System.out.println(myFile.getAbsolutePath() +
                " cannot be set to read-only.");
        }
    }
}

/*
Output:
C:\MyFile.txt is now read-only.
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.toString Method

Displays a human readable representation of a file or folder.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

A human readable representation of a file or folder. The value displayed is the path of the file or folder.

See Also

Reference

File Class

Concepts

File Members

java.io Package

File.toURL Method

Converts a file or folder into a URL (Uniform Resource Locator).

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.net.URL toURL() throws java.net.MalformedURLException;
```

Return Value

A URL representing the file or folder.

Example

The following example prints the URL of a file.

```
// file_tourl.jsl

import java.io.*;
import java.net.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Open the file C:\MyFile.txt.
            File myFile = new File("C:\\Windows\\..\\MyFile.txt");
            URL url = myFile.toURL();

            // If the file exists, display it's full path.
            if (myFile.exists())
            {
                System.out.println(url.toString() + " exists!");
            }
            else
            {
                System.out.println(url.toString() + " does NOT exist!");
            }
        }
        catch (MalformedURLException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
file:/C:/MyFile.txt exists!
*/
```

See Also

Reference

File Class

Concepts

File Members

java.io Package

FileDescriptor Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final class java.io.FileDescriptor
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.io.FileDescriptor

See Also

Concepts

FileDescriptor Members

java.io Package

FileDescriptor Members

The following tables list the members exposed by the FileDescriptor type.

Public Constructors

Name	Description
FileDescriptor	

Public Fields

Name	Description
err	
in	
out	

Public Methods

Name	Description
Equals	(inherited from Object)
Finalize	
GetHashCode	(inherited from **Object**)
GetType	(inherited from **Object**)
sync	
valid	

See Also

Reference

FileDescriptor Class

Concepts

java.io Package

FileDescriptor Fields

Public Fields

Name	Description
err	
in	
out	

See Also

Reference

FileDescriptor Class

Concepts

java.io Package

FileDescriptor.err Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final java.io.FileDescriptor err;
```

See Also

Reference

FileDescriptor Class

Concepts

FileDescriptor Members

java.io Package

FileDescriptor.in Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final java.io.FileDescriptor in;
```

See Also

Reference

FileDescriptor Class

Concepts

FileDescriptor Members

java.io Package

FileDescriptor.out Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final java.io.FileDescriptor out;
```

See Also

Reference

FileDescriptor Class

Concepts

FileDescriptor Members

java.io Package

FileDescriptor Constructor

Initializes a new instance of the FileDescriptor Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileDescriptor();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

FileDescriptor Class

Concepts

FileDescriptor Members

java.io Package

FileDescriptor Methods

Public Methods

Name	Description
equals	(inherited from Object)
finalize	
hashCode	(inherited from Object)
getClass	(inherited from Object)
sync	
valid	

See Also

Reference

FileDescriptor Class

Concepts

java.io Package

FileDescriptor.clone Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

See Also

Reference

FileDescriptor Class

Concepts

FileDescriptor Members

java.io Package

FileDescriptor.finalize Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void finalize() throws java.io.IOException;
```

See Also

Reference

FileDescriptor Class

Concepts

FileDescriptor Members

java.io Package

FileDescriptor.sync Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void sync() throws java.io.SyncFailedException;
```

See Also

Reference

FileDescriptor Class

Concepts

FileDescriptor Members

java.io Package

FileDescriptor.valid Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean valid();
```

See Also

Reference

FileDescriptor Class

Concepts

FileDescriptor Members

java.io Package

FileFilter Interface

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public interface java.io.FileFilter
```

See Also

Concepts

FileFilter Members

java.io Package

FileFilter Members

The following tables list the members exposed by the FileFilter type.

Public Methods

Name	Description
accept	

See Also

Reference

FileFilter Interface

Concepts

java.io Package

FileFilter Methods

Public Methods

Name	Description
accept	

See Also

Reference

FileFilter Interface

Concepts

java.io Package

FileFilter.accept Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean accept(
    java.io.File pathname);
```

Parameters
pathname

See Also
Reference
FileFilter Interface
Concepts
FileFilter Members
java.io Package

FileInputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.FileInputStream
    extends java.io.InputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

   java.io.FileInputStream

See Also

Concepts

FileInputStream Members

java.io Package

FileInputStream Members

The following tables list the members exposed by the FileInputStream type.

Public Constructors

Name	Description
FileInputStream	Overloaded.

Public Methods

Name	Description
available	Overridden.
close	Overridden.
equals	(inherited from Object)
finalize	Overridden.
getFD	
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	(inherited from InputStream)
skip	Overridden.
toString	(inherited from InputStream)

See Also

Reference

FileInputStream Class

Concepts

java.io Package

FileInputStream Constructor

Overload List

Name	Description
FileInputStream (File)	
FileInputStream (FileDescriptor)	
FileInputStream (String)	

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream Constructor (File)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileInputStream(
    java.io.File file) throws java.io.FileNotFoundException;
```

Parameters

file

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream Constructor (FileDescriptor)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileInputStream(
    java.io.FileDescriptor fd);
```

Parameters

fd

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileInputStream(
    java.lang.String path) throws java.io.FileNotFoundException;
```

Parameters

path

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream Methods

Public Methods

Name	Description
available	Overridden.
close	Overridden.
equals	(inherited from Object)
finalize	Overridden.
getFD	
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	(inherited from InputStream)
skip	Overridden.
toString	(inherited from InputStream)

See Also

Reference

FileInputStream Class

Concepts

java.io Package

FileInputStream.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int available() throws java.io.IOException;
```

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream.finalize Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void finalize() throws java.io.IOException;
```

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream.getFD Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final java.io.FileDescriptor getFD() throws java.io.IOException;
```

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream.read Method

Overload List

Name	Description
FileInputStream.read ()	
FileInputStream.read (byte[])	
FileInputStream.read (byte[], int, int)	

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream.read Method (SByte [])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] buffer) throws java.io.IOException;
```

Parameters

buffer

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream.read Method (SByte[], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] buffer,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buffer

offset

count

See Also

Reference

FileInputStream Class

Concepts

FileInputStream Members

java.io Package

FileInputStream.skip Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long count) throws java.io.IOException;
```

Parameters

count

See Also
Reference
FileInputStream Class
Concepts
FileInputStream Members
java.io Package

FilenameFilter Interface

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public interface java.io.FilenameFilter
```

See Also

Concepts

FilenameFilter Members

java.io Package

FilenameFilter Members

The following tables list the members exposed by the FilenameFilter type.

Public Methods

Name	Description
accept	

See Also

Reference

FilenameFilter Interface

Concepts

java.io Package

FilenameFilter Methods

Public Methods

Name	Description
accept	

See Also

Reference

FilenameFilter Interface

Concepts

java.io Package

FilenameFilter.accept Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean accept(
    java.io.File dir,
    java.lang.String name);
```

Parameters

dir

name

See Also

Reference

FilenameFilter Interface

Concepts

FilenameFilter Members

java.io Package

FileNotFoundException Class

The exception that is thrown when an attempt is made to access a file that does not exist.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.FileNotFoundException
    extends java.io.IOException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.FileNotFoundException

See Also

Concepts

FileNotFoundException Members

java.io Package

FileNotFoundException Members

The exception that is thrown when an attempt is made to access a file that does not exist.

The following tables list the members exposed by the FileNotFoundException type.

Public Constructors

Name	Description
FileNotFoundException	Overloaded. Initializes a new instance of a FileNotFoundException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FileNotFoundException Class

Concepts

java.io Package

FileNotFoundException Constructor

Initializes a new instance of a FileNotFoundException object.

Overload List

Name	Description
FileNotFoundException ()	Initializes a new instance of a FileNotFoundException object.
FileNotFoundException (String)	Initializes a new instance of a FileNotFoundException object with the given message.
FileNotFoundException (SerializationInfo, StreamingContext)	Initializes a new instance of a FileNotFoundException object during deserialization.
FileNotFoundException (String, Exception)	Initializes a new instance of a FileNotFoundException object with the given message and inner exception.

See Also

Reference

FileNotFoundException Class

Concepts

FileNotFoundException Members

java.io Package

FileNotFoundException Constructor ()

Initializes a new instance of a FileNotFoundException object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileNotFoundException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

FileNotFoundException Class

Concepts

FileNotFoundException Members

java.io Package

FileNotFoundException Constructor (String)

Initializes a new instance of a FileNotFoundException object with the given message.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileNotFoundException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also

Reference

FileNotFoundException Class

Concepts

FileNotFoundException Members

java.io Package

FileNotFoundException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a FileNotFoundException object during deserialization.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.FileNotFoundException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
FileNotFoundException Class
Concepts
FileNotFoundException Members
java.io Package

FileNotFoundException Constructor (String, Exception)

Initializes a new instance of a FileNotFoundException object with the given message and inner exception.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileNotFoundException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the FileNotFoundException.

See Also

Reference

FileNotFoundException Class

Concepts

FileNotFoundException Members

java.io Package

FileNotFoundException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FileNotFoundException Class

Concepts

java.io Package

FileNotFoundException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

FileNotFoundException Class

Concepts

java.io Package

FileOutputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.FileOutputStream
    extends java.io.OutputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.OutputStream

   java.io.FileOutputStream

See Also

Concepts

FileOutputStream Members

java.io Package

FileOutputStream Members

The following tables list the members exposed by the FileOutputStream type.

Public Constructors

Name	Description
FileOutputStream	Overloaded.

Public Methods

Name	Description
close	Overridden.
equals	(inherited from Object)
finalize	Overridden.
flush	(inherited from OutputStream)
getFD	
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

See Also

Reference

FileOutputStream Class

Concepts

java.io Package

FileOutputStream Constructor

Overload List

Name	Description
FileOutputStream (File)	
FileOutputStream (FileDescriptor)	
FileOutputStream (String)	
FileOutputStream (String, boolean)	

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream Constructor (File)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileOutputStream(
    java.io.File file) throws java.io.IOException;
```

Parameters

file

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream Constructor (FileDescriptor)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileOutputStream(
    java.io.FileDescriptor fd);
```

Parameters

fd

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileOutputStream(
    java.lang.String path) throws java.io.IOException;
```

Parameters

path

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream Constructor (String, Boolean)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileOutputStream(
    java.lang.String path,
    boolean append) throws java.io.IOException;
```

Parameters

path

append

See Also
Reference
FileOutputStream Class
Concepts
FileOutputStream Members
java.io Package

FileOutputStream Methods

Public Methods

Name	Description
close	Overridden.
equals	(inherited from Object)
finalize	Overridden.
flush	(inherited from OutputStream)
getFD	
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

See Also

Reference

FileOutputStream Class

Concepts

java.io Package

FileOutputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream.finalize Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void finalize() throws java.io.IOException;
```

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream.getFD Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final java.io.FileDescriptor getFD() throws java.io.IOException;
```

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream.write Method

Overload List

Name	Description
FileOutputStream.write (int)	
FileOutputStream.write (byte[])	
FileOutputStream.write (byte[], int, int)	

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream.write Method (SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] data) throws java.io.IOException;
```

Parameters

data

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileOutputStream.write Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] buffer,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buffer

offset

count

See Also

Reference

FileOutputStream Class

Concepts

FileOutputStream Members

java.io Package

FileReader Class

Provides an implementation of the abstract Reader class used to provide a character stream for files.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.FileReader
    extends java.io.InputStreamReader
```

Inheritance Hierarchy

java.lang.Object

  java.io.Reader

    java.io.InputStreamReader

      java.io.FileReader

See Also

Concepts

FileReader Members

java.io Package

FileReader Members

Provides an implementation of the abstract Reader class used to provide a character stream for files.

The following tables list the members exposed by the FileReader type.

Public Constructors

Name	Description
FileReader	Overloaded. Initializes a new instance of a FileReader object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader.(inherited from Reader)

Public Methods

Name	Description
close	Closes the underlying InputStream and releases all internal buffers. (inherited from InputStreamReader)
equals	(inherited from Object)
getEncoding	Returns the character encoding of the underlying InputStream object. (inherited from InputStreamReader)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Sets the mark to the next character of the characters to be read. (inherited from Reader)
markSupported	Determines whether the current position of the character can be marked. (inherited from Reader)
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Reads the next character or characters from the underlying InputStream. (inherited from InputStreamReader)
ready	Determines whether the underlying InputStream object is ready for reading. (inherited from InputStreamReader)
reset	Resets the reader for the characters such that the next character read is the character that is marked. (inherited from Reader)
skip	Skips over the next count characters of the characters being read or until the end of the array, whichever is less. (inherited from Reader)
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FileReader Class

Concepts

java.io Package

FileReader Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader. (inherited from Reader)

See Also

Reference

FileReader Class

Concepts

java.io Package

FileReader Constructor

Initializes a new instance of a FileReader object.

Overload List

Name	Description
FileReader (File)	Initializes a new instance of a FileReader object using the file represented by a given File.
FileReader (FileDescriptor)	Initializes a new instance of a FileReader object using the file represented by the FileDescriptor.
FileReader (String)	Initializes a new instance of a FileReader object using the given file name.

See Also

Reference

FileReader Class

Concepts

FileReader Members

java.io Package

FileReader Constructor (File)

Initializes a new instance of a FileReader object using the file represented by a given File.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileReader(
    java.io.File file) throws java.io.FileNotFoundException;
```

Parameters

file

A File object representing the file to be read.

See Also

Reference

FileReader Class

Concepts

FileReader Members

java.io Package

FileReader Constructor (FileDescriptor)

Initializes a new instance of a FileReader object using the file represented by the FileDescriptor.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileReader(
    java.io.FileDescriptor fd);
```

Parameters

fd

A FileDescriptor object representing the file to be read.

See Also

Reference

FileReader Class

Concepts

FileReader Members

java.io Package

FileReader Constructor (String)

Initializes a new instance of a FileReader object using the given file name.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileReader(
    java.lang.String filename) throws java.io.FileNotFoundException;
```

Parameters

filename

The name of the file to be read.

See Also

Reference

FileReader Class

Concepts

FileReader Members

java.io Package

FileReader Methods

Public Methods

Name	Description
close	Closes the underlying InputStream and releases all internal buffers. (inherited from InputStreamReader)
equals	(inherited from Object)
getEncoding	Returns the character encoding of the underlying InputStream object. (inherited from InputStreamReader)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Sets the mark to the next character of the characters to be read. (inherited from Reader)
markSupported	Determines whether the current position of the character can be marked. (inherited from Reader)
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Reads the next character or characters from the underlying InputStream. (inherited from InputStreamReader)
ready	Determines whether the underlying InputStream object is ready for reading. (inherited from InputStreamReader)
reset	Resets the reader for the characters such that the next character read is the character that is marked. (inherited from Reader)
skip	Skips over the next count characters of the characters being read or until the end of the array, whichever is less. (inherited from Reader)
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FileReader Class

Concepts

java.io Package

FileWriter Class

Provides an implementation of the abstract Writer class specific to writing files.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.FileWriter
    extends java.io.OutputStreamWriter
```

Inheritance Hierarchy

java.lang.Object

  java.io.Writer

    java.io.OutputStreamWriter

      java.io.FileWriter

See Also

Concepts

FileWriter Members

java.io Package

FileWriter Members

Provides an implementation of the abstract Writer class specific to writing files.

The following tables list the members exposed by the FileWriter type.

Public Constructors

Name	Description
FileWriter	Overloaded. Initializes a new instance of a FileWriter object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer.(inherited from Writer)

Public Methods

Name	Description
close	Closes the underlying OutputStream and releases all internal buffers. (inherited from OutputStreamWriter)
equals	(inherited from Object)
flush	Flushes the underlying OutputStream object. (inherited from OutputStreamWriter)
getEncoding	Returns the character encoding being used on this instance. (inherited from OutputStreamWriter)
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Writes the next character or characters to the underlying OutputStream. (inherited from OutputStreamWriter)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FileWriter Class

Concepts

java.io Package

FileWriter Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer. (inherited from Writer)

See Also

Reference

FileWriter Class

Concepts

java.io Package

FileWriter Constructor

Initializes a new instance of a FileWriter object.

Overload List

Name	Description
FileWriter (File)	Initializes a new instance of a FileWriter object using the file represented by a given File.
FileWriter (FileDescriptor)	Initializes a new instance of a FileWriter object using the file represented by the FileDescriptor.
FileWriter (String)	Initializes a new instance of a FileWriter object using the given file name.
FileWriter (String, boolean)	Initializes a new instance of a FileWriter object using the given file name. The file can either be app ended or overwritten.

See Also

Reference

FileWriter Class

Concepts

FileWriter Members

java.io Package

FileWriter Constructor (File)

Initializes a new instance of a FileWriter object using the file represented by a given File.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileWriter(
    java.io.File file) throws java.io.IOException;
```

Parameters

file

A File object representing the file to be written to.

See Also

Reference

FileWriter Class

Concepts

FileWriter Members

java.io Package

FileWriter Constructor (FileDescriptor)

Initializes a new instance of a FileWriter object using the file represented by the FileDescriptor.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileWriter(
    java.io.FileDescriptor fd);
```

Parameters

fd

A FileDescriptor object representing the file to be written to.

See Also

Reference

FileWriter Class

Concepts

FileWriter Members

java.io Package

FileWriter Constructor (String)

Initializes a new instance of a FileWriter object using the given file name.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileWriter(
    java.lang.String filename) throws java.io.IOException;
```

Parameters

filename

The name of the file to be read.

See Also

Reference

FileWriter Class

Concepts

FileWriter Members

java.io Package

FileWriter Constructor (String, Boolean)

Initializes a new instance of a FileWriter object using the given file name. The file can either be appended or overwritten.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FileWriter(
    java.lang.String filename,
    boolean append) throws java.io.IOException;
```

Parameters

filename

The name of the file to be read.

append

true to append to the end of the file; false to replace the file.

See Also

Reference

FileWriter Class

Concepts

FileWriter Members

java.io Package

FileWriter Methods

Public Methods

Name	Description
close	Closes the underlying OutputStream and releases all internal buffers. (inherited from OutputStreamWriter)
equals	(inherited from Object)
flush	Flushes the underlying OutputStream object. (inherited from OutputStreamWriter)
getEncoding	Returns the character encoding being used on this instance. (inherited from OutputStreamWriter)
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Writes the next character or characters to the underlying OutputStream. (inherited from OutputStreamWriter)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FileWriter Class

Concepts

java.io Package

FilterInputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.FilterInputStream
    extends java.io.InputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

   java.io.FilterInputStream

     Derived Classes

See Also

Concepts

FilterInputStream Members

java.io Package

FilterInputStream Members

The following tables list the members exposed by the FilterInputStream type.

Public Constructors

Name	Description
FilterInputStream	

Public Fields

Name	Description
in	

Public Methods

Name	Description
available	Overridden.
close	Overridden.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden.
markSupported	Overridden.
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden.
skip	Overridden.
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FilterInputStream Class

Concepts

java.io Package

FilterInputStream Fields

Public Fields

Name	Description
in	

See Also

Reference

FilterInputStream Class

Concepts

java.io Package

FilterInputStream.in Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.InputStream in;
```

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterInputStream Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.FilterInputStream(
    java.io.InputStream in);
```

Parameters

in

See Also
Reference
FilterInputStream Class
Concepts
FilterInputStream Members
java.io Package

FilterInputStream Methods

Public Methods

Name	Description
available	Overridden.
close	Overridden.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden.
markSupported	Overridden.
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden.
skip	Overridden.
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FilterInputStream Class

Concepts

java.io Package

FilterInputStream.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int available() throws java.io.IOException;
```

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterInputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterInputStream.mark Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void mark(
    int readlimit);
```

Parameters
readlimit

See Also
Reference
FilterInputStream Class
Concepts
FilterInputStream Members
java.io Package

FilterInputStream.markSupported Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterInputStream.read Method

Overload List

Name	Description
FilterInputStream.read ()	
FilterInputStream.read (byte[])	
FilterInputStream.read (byte[], int, int)	

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterInputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterInputStream.read Method (SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterInputStream.read Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterInputStream.reset Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void reset() throws java.io.IOException;
```

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterInputStream.skip Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long n) throws java.io.IOException;
```

Parameters

n

See Also

Reference

FilterInputStream Class

Concepts

FilterInputStream Members

java.io Package

FilterOutputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.FilterOutputStream
    extends java.io.OutputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.OutputStream

  java.io.FilterOutputStream

    Derived Classes

See Also

Concepts

FilterOutputStream Members

java.io Package

FilterOutputStream Members

The following tables list the members exposed by the FilterOutputStream type.

Public Constructors

Name	Description
FilterOutputStream	

Public Fields

Name	Description
out	

Public Methods

Name	Description
close	Overridden.
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FilterOutputStream Class

Concepts

java.io Package

FilterOutputStream Fields

Public Fields

Name	Description
out	

See Also

Reference

FilterOutputStream Class

Concepts

java.io Package

FilterOutputStream.out Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.OutputStream out;
```

See Also

Reference

FilterOutputStream Class

Concepts

FilterOutputStream Members

java.io Package

FilterOutputStream Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.FilterOutputStream(
    java.io.OutputStream out);
```

Parameters

out

See Also

Reference

FilterOutputStream Class

Concepts

FilterOutputStream Members

java.io Package

FilterOutputStream Methods

Public Methods

Name	Description
close	Overridden.
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FilterOutputStream Class

Concepts

java.io Package

FilterOutputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

FilterOutputStream Class

Concepts

FilterOutputStream Members

java.io Package

FilterOutputStream.flush Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush() throws java.io.IOException;
```

See Also

Reference

FilterOutputStream Class

Concepts

FilterOutputStream Members

java.io Package

FilterOutputStream.write Method

Overload List

Name	Description
FilterOutputStream.write (int)	
FilterOutputStream.write (byte[])	
FilterOutputStream.write (byte[], int, int)	

See Also

Reference

FilterOutputStream Class

Concepts

FilterOutputStream Members

java.io Package

FilterOutputStream.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

FilterOutputStream Class

Concepts

FilterOutputStream Members

java.io Package

FilterOutputStream.write Method (SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

FilterOutputStream Class

Concepts

FilterOutputStream Members

java.io Package

FilterOutputStream.write Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

FilterOutputStream Class

Concepts

FilterOutputStream Members

java.io Package

FilterReader Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.io.FilterReader
    extends java.io.Reader
```

Inheritance Hierarchy

java.lang.Object

  java.io.Reader

   java.io.FilterReader

    java.io.PushbackReader

See Also

Concepts

FilterReader Members

java.io Package

FilterReader Members

The following tables list the members exposed by the FilterReader type.

Public Constructors

Name	Description
FilterReader	

Public Fields

Name	Description
in	
lock	The object used to synchronize operations on the reader.(inherited from Reader)

Public Methods

Name	Description
close	Overridden.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden.
markSupported	Overridden.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden.
ready	Overridden.
reset	Overridden.
skip	Overridden.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FilterReader Class

Concepts

java.io Package

FilterReader Fields

Public Fields

Name	Description
in	
lock	The object used to synchronize operations on the reader. (inherited from Reader)

See Also

Reference

FilterReader Class

Concepts

java.io Package

FilterReader.in Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.Reader in;
```

See Also

Reference

FilterReader Class

Concepts

FilterReader Members

java.io Package

FilterReader Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.FilterReader(
    java.io.Reader in);
```

Parameters

in

See Also
Reference
FilterReader Class
Concepts
FilterReader Members
java.io Package

FilterReader Methods

Public Methods

Name	Description
close	Overridden.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden.
markSupported	Overridden.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden.
ready	Overridden.
reset	Overridden.
skip	Overridden.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FilterReader Class

Concepts

java.io Package

FilterReader.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

FilterReader Class

Concepts

FilterReader Members

java.io Package

FilterReader.mark Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void mark(
    int readlimit) throws java.io.IOException;
```

Parameters
readlimit

See Also
Reference
FilterReader Class
Concepts
FilterReader Members
java.io Package

FilterReader.markSupported Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

See Also

Reference

FilterReader Class

Concepts

FilterReader Members

java.io Package

FilterReader.read Method

Overload List

Name	Description
FilterReader.read ()	
FilterReader.read (char[])	Reads the characters in the array starting at an offset for the length specified and stores them in the provided array of characters.
FilterReader.read (char[], int, int)	

See Also

Reference

FilterReader Class

Concepts

FilterReader Members

java.io Package

FilterReader.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

See Also

Reference

FilterReader Class

Concepts

FilterReader Members

java.io Package

FilterReader.read Method (Char[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also
Reference
FilterReader Class
Concepts
FilterReader Members
java.io Package

FilterReader.ready Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean ready() throws java.io.IOException;
```

See Also

Reference

FilterReader Class

Concepts

FilterReader Members

java.io Package

FilterReader.reset Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void reset() throws java.io.IOException;
```

See Also

Reference

FilterReader Class

Concepts

FilterReader Members

java.io Package

FilterReader.skip Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long n) throws java.io.IOException;
```

Parameters

n

See Also
Reference
FilterReader Class
Concepts
FilterReader Members
java.io Package

FilterWriter Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.io.FilterWriter
    extends java.io.Writer
```

Inheritance Hierarchy

java.lang.Object

  java.io.Writer

   java.io.FilterWriter

See Also

Concepts

FilterWriter Members

java.io Package

FilterWriter Members

The following tables list the members exposed by the FilterWriter type.

Public Constructors

Name	Description
FilterWriter	

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer.(inherited from Writer)
out	

Public Methods

Name	Description
close	Overridden.
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FilterWriter Class

Concepts

java.io Package

FilterWriter Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer. (inherited from Writer)
out	

See Also

Reference

FilterWriter Class

Concepts

java.io Package

FilterWriter.out Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.Writer out;
```

See Also

Reference

FilterWriter Class

Concepts

FilterWriter Members

java.io Package

FilterWriter Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.FilterWriter(
    java.io.Writer out);
```

Parameters

out

See Also

Reference

FilterWriter Class

Concepts

FilterWriter Members

java.io Package

FilterWriter Methods

Public Methods

Name	Description
close	Overridden.
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FilterWriter Class

Concepts

java.io Package

FilterWriter.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

FilterWriter Class

Concepts

FilterWriter Members

java.io Package

FilterWriter.flush Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush() throws java.io.IOException;
```

See Also

Reference

FilterWriter Class

Concepts

FilterWriter Members

java.io Package

FilterWriter.write Method

Overload List

Name	Description
FilterWriter.write (char[])	Places the given array of characters into the internal buffer holding the characters.
FilterWriter.write (int)	
FilterWriter.write (String)	Places the given string into the internal buffer holding characters.
FilterWriter.write (char[], int, int)	
FilterWriter.write (String, int, int)	

See Also

Reference

FilterWriter Class

Concepts

FilterWriter Members

java.io Package

FilterWriter.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int oneChar) throws java.io.IOException;
```

Parameters
oneChar

See Also
Reference
FilterWriter Class
Concepts
FilterWriter Members
java.io Package

FilterWriter.write Method (Char[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    char[] b,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

b

offset

count

See Also

Reference

FilterWriter Class

Concepts

FilterWriter Members

java.io Package

FilterWriter.write Method (String, Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

str

offset

count

See Also

Reference

FilterWriter Class

Concepts

FilterWriter Members

java.io Package

InputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.io.InputStream
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.io.InputStream

Derived Classes

See Also

Concepts

InputStream Members

java.io Package

InputStream Members

The following tables list the members exposed by the InputStream type.

Public Constructors

Name	Description
InputStream	

Public Methods

Name	Description
available	
close	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	
markSupported	
read	Overloaded.
reset	
skip	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InputStream Class

Concepts

java.io Package

InputStream Constructor

Initializes a new instance of the InputStream Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InputStream();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStream Methods

Public Methods

Name	Description
available	
close	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	
markSupported	
read	Overloaded.
reset	
skip	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InputStream Class

Concepts

java.io Package

InputStream.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int available() throws java.io.IOException;
```

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStream.mark Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void mark(
    int readlimit);
```

Parameters
readlimit

See Also
Reference
InputStream Class
Concepts
InputStream Members
java.io Package

InputStream.markSupported Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStream.read Method

Overload List

Name	Description
InputStream.read ()	
InputStream.read (byte[])	
InputStream.read (byte[], int, int)	

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int read() throws java.io.IOException;
```

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStream.read Method (SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStream.read Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] b,
    int off,
    int count) throws java.io.IOException;
```

Parameters

b

off

count

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStream.reset Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void reset() throws java.io.IOException;
```

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStream.skip Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long count) throws java.io.IOException;
```

Parameters

count

See Also

Reference

InputStream Class

Concepts

InputStream Members

java.io Package

InputStreamReader Class

Provides an implementation of the abstract Reader class specific to reading characters in an InputStream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.InputStreamReader
    extends java.io.Reader
```

Example

The following example demonstrates the close, getEncoding, read, and ready methods of the InputStreamReader class.

```
// inputstreamreader_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a InputStreamReader object
            // attached to a StringBufferInputStream.
            String s = "StringBufferInputStream";
            StringBufferInputStream buf = new StringBufferInputStream(s);
            InputStreamReader reader = new InputStreamReader(buf);

            // Read from the input stream.
            char[] arr = new char[s.length()];
            if (reader.ready())
            {
                reader.read(arr, 0, arr.length);
            }
            System.out.println(arr);

            // Display the encoding of the input stream.
            System.out.println("encoding: " + reader.getEncoding());

            // Close the InputStreamReader and the
            // StringBufferInputStream objects.
            buf.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
StringBufferInputStream
encoding: Cp1252
*/
```

Inheritance Hierarchy

java.lang.Object

java.io.Reader

java.io.InputStreamReader

java.io.FileReader

See Also

Concepts

InputStreamReader Members

java.io Package

InputStreamReader Members

Provides an implementation of the abstract Reader class specific to reading characters in an InputStream object.

The following tables list the members exposed by the InputStreamReader type.

Public Constructors

Name	Description
InputStreamReader	Overloaded. Initializes a new instance of an InputStreamReader object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader.(inherited from Reader)

Public Methods

Name	Description
close	Overridden. Closes the underlying InputStream and releases all internal buffers.
equals	(inherited from Object)
getEncoding	Returns the character encoding of the underlying InputStream object.
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Sets the mark to the next character of the characters to be read. (inherited from Reader)
markSupported	Determines whether the current position of the character can be marked. (inherited from Reader)
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the underlying InputStream.
ready	Overridden. Determines whether the underlying InputStream object is ready for reading.
reset	Resets the reader for the characters such that the next character read is the character that is marked. (inherited from Reader)
skip	Skips over the next count characters of the characters being read or until the end of the array, whichever is less. (inherited from Reader)
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InputStreamReader Class
Concepts
java.io Package

InputStreamReader Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader. (inherited from Reader)

See Also

Reference

InputStreamReader Class

Concepts

java.io Package

InputStreamReader Constructor

Initializes a new instance of an InputStreamReader object.

Overload List

Name	Description
InputStreamReader (InputStream)	Initializes a new instance of an InputStreamReader object using the given InputStream object as the underlying stream.
InputStreamReader (InputStream, String)	Initializes a new instance of an InputStreamReader object using the given InputStream object as the underlying stream. Also specifies the character encoding of the underlying stream.

See Also

Reference

InputStreamReader Class

Concepts

InputStreamReader Members

java.io Package

InputStreamReader Constructor (InputStream)

Initializes a new instance of an InputStreamReader object using the given InputStream object as the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InputStreamReader(
    java.io.InputStream in);
```

Parameters

in

The underlying input stream to read from.

See Also

Reference

InputStreamReader Class

Concepts

InputStreamReader Members

java.io Package

InputStreamReader Constructor (InputStream, String)

Initializes a new instance of an InputStreamReader object using the given InputStream object as the underlying stream. Also specifies the character encoding of the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InputStreamReader(
    java.io.InputStream in,
    java.lang.String enc) throws java.io.UnsupportedEncodingException;
```

Parameters

in

The underlying input stream to read from.

enc

The character encoding of the input stream (such as Unicode).

See Also

Reference

InputStreamReader Class

Concepts

InputStreamReader Members

java.io Package

InputStreamReader Methods

Public Methods

Name	Description
close	Overridden. Closes the underlying InputStream and releases all internal buffers.
equals	(inherited from Object)
getEncoding	Returns the character encoding of the underlying InputStream object.
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Sets the mark to the next character of the characters to be read. (inherited from Reader)
markSupported	Determines whether the current position of the character can be marked. (inherited from Reader)
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the underlying InputStream.
ready	Overridden. Determines whether the underlying InputStream object is ready for reading.
reset	Resets the reader for the characters such that the next character read is the character that is marked. (inherited from Reader)
skip	Skips over the next count characters of the characters being read or until the end of the array, whichever is less. (inherited from Reader)
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InputStreamReader Class

Concepts

java.io Package

InputStreamReader.close Method

Closes the underlying InputStream and releases all internal buffers.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close a InputStreamReader object.

```
// inputstreamreader_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a InputStreamReader object
            // attached to a StringBufferInputStream.
            String s = "StringBufferInputStream";
            StringBufferInputStream buf = new StringBufferInputStream(s);
            InputStreamReader reader = new InputStreamReader(buf);

            // Read from the input stream.
            int charRead;
            while ((charRead = reader.read()) >= 0)
            {
                System.out.print((char)charRead);
            }

            // Close the InputStreamReader and the
            // StringBufferInputStream objects.
            buf.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
StringBufferInputStream
*/
```

See Also

Reference

InputStreamReader Class

Concepts

InputStreamReader Members

java.io Package

InputStreamReader.getEncoding Method

Returns the character encoding of the underlying InputStream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getEncoding();
```

Return Value

The character encoding of the underlying InputStream object.

Example

The following example demonstrates how to determine the encoding of the underlying input stream.

```
// inputstreamreader_getencoding.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a InputStreamReader object
            // attached to a StringBufferInputStream.
            String s = "StringBufferInputStream";
            StringBufferInputStream buf = new StringBufferInputStream(s);
            InputStreamReader reader = new InputStreamReader(buf);

            // Read from the input stream.
            int charRead;
            while ((charRead = reader.read()) >= 0)
            {
                System.out.print((char)charRead);
            }
            System.out.println();

            // Display the encoding of the input stream.
            System.out.println("encoding: " + reader.getEncoding());

            // Close the InputStreamReader and the
            // StringBufferInputStream objects.
            buf.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
StringBufferInputStream
encoding: Cp1252
*/
```

See Also

Reference
InputStreamReader Class
Concepts
InputStreamReader Members
java.io Package

InputStreamReader.read Method

Reads the next character or characters from the underlying InputStream.

Overload List

Name	Description
InputStreamReader.read ()	Reads the next character from the underlying InputStream.
InputStreamReader.read (char[])	Reads the characters in the array starting at an offset for the length specified and stores them in the provided array of characters.
InputStreamReader.read (char[], int, int)	Reads count characters from the underlying InputStream object, starting from an offset, and stores the characters in the provided buffer.

See Also

Reference

InputStreamReader Class

Concepts

InputStreamReader Members

java.io Package

InputStreamReader.read Method ()

Reads the next character from the underlying InputStream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

Return Value

The numeric value of the character read using the character encoding of the system (such as UTF-8).

Example

The following example demonstrates how to read one character at a time from an input stream.

```
// inputstreamreader_read.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a InputStreamReader object
            // attached to a StringBufferInputStream.
            String s = "StringBufferInputStream";
            StringBufferInputStream buf = new StringBufferInputStream(s);
            InputStreamReader reader = new InputStreamReader(buf);

            // Read from the input stream.
            int charRead;
            while ((charRead = reader.read()) >= 0)
            {
                System.out.print((char)charRead);
            }
            System.out.println();

            // Close the InputStreamReader and the
            // StringBufferInputStream objects.
            buf.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
StringBufferInputStream
*/
```

See Also

Reference

InputStreamReader Class

Concepts

InputStreamReader Members

InputStreamReader.read Method (Char[ ], Int32, Int32)

Reads count characters from the underlying InputStream object, starting from an offset, and stores the characters in the provided buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    char[] buf,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buf

A buffer to store the characters read from the underlying InputStream.

offset

An offset into the array of characters. This value represents the index of the first character in the array to be written. This value must be greater than zero.

count

The number of characters to be read from the underlying stream. This value plus the offset must be less than the overall length of the buffer.

Return Value

The number of characters read from the underlying InputStream.

Example

The following example demonstrates how to read from an input stream and store the data read into an array.

```
// inputstreamreader_read_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a InputStreamReader object
            // attached to a StringBufferInputStream.
            String s = "StringBufferInputStream";
            StringBufferInputStream buf = new StringBufferInputStream(s);
            InputStreamReader reader = new InputStreamReader(buf);

            // Read from the input stream.
            char[] arr = new char[s.length()];
            reader.read(arr, 0, arr.length);
            System.out.println(arr);

            // Close the InputStreamReader and the
            // StringBufferInputStream objects.
            buf.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
```

```
        }
    }
}

/*
Output:
StringBufferInputStream
*/
```

See Also

Reference

InputStreamReader Class

Concepts

InputStreamReader Members

java.io Package

InputStreamReader.ready Method

Determines whether the underlying InputStream object is ready for reading.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean ready() throws java.io.IOException;
```

Return Value

true if the underlying InputStream object is ready for reading; false otherwise.

Example

The following example demonstrates how to determine if an input stream is ready to be read from.

```
// inputstreamreader_ready.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a InputStreamReader object
            // attached to a StringBufferInputStream.
            String s = "StringBufferInputStream";
            StringBufferInputStream buf = new StringBufferInputStream(s);
            InputStreamReader reader = new InputStreamReader(buf);

            // Read from the input stream.
            char[] arr = new char[s.length()];
            if (reader.ready())
            {
                reader.read(arr, 0, arr.length);
            }
            System.out.println(arr);

            // Close the InputStreamReader and the
            // StringBufferInputStream objects.
            buf.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
StringBufferInputStream
*/
```

See Also

Reference

InputStreamReader Class

Concepts

InputStreamReader Members

InterruptedIOException Class

The exception that is thrown when a timeout has occurred or there was an interruption during an I/O operation.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.InterruptedIOException
    extends java.io.IOException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.InterruptedIOException

See Also

Concepts

InterruptedIOException Members

java.io Package

InterruptedIOException Members

The exception that is thrown when a timeout has occurred or there was an interruption during an I/O operation.

The following tables list the members exposed by the InterruptedIOException type.

Public Constructors

Name	Description
InterruptedIOException	Overloaded. Initializes a new instance of an InterruptedIOException object.

Public Fields

Name	Description
bytesTransferred	Contains the number of bytes that were successfully transferred before the I/O operation was interrupted.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden. Retrieves the data to be serialized.
getClass	(inherited from Object)

clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InterruptedIOException Class

Concepts

java.io Package

InterruptedIOException Fields

Public Fields

Name	Description
bytesTransferred	Contains the number of bytes that were successfully transferred before the I/O operation was interrupted.

See Also

Reference

InterruptedIOException Class

Concepts

java.io Package

InterruptedIOException.bytesTransferred Field

Contains the number of bytes that were successfully transferred before the I/O operation was interrupted.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int bytesTransferred;
```

See Also

Reference

InterruptedIOException Class

Concepts

InterruptedIOException Members

java.io Package

InterruptedIOException Constructor

Initializes a new instance of an InterruptedIOException object.

Overload List

Name	Description
InterruptedIOException ()	Initializes a new instance of an InterruptedIOException object.
InterruptedIOException (String)	Initializes a new instance of an InterruptedIOException object with the given message.
InterruptedIOException (SerializationInfo, StreamingContext)	Initializes a new instance of an InterruptedIOException object during deserialization.
InterruptedIOException (String, Exception)	Initializes a new instance of an InterruptedIOException object with the given message and inner exception.

See Also

Reference

InterruptedIOException Class

Concepts

InterruptedIOException Members

java.io Package

InterruptedIOException Constructor ()

Initializes a new instance of an InterruptedIOException object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InterruptedIOException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

InterruptedIOException Class

Concepts

InterruptedIOException Members

java.io Package

InterruptedIOException Constructor (String)

Initializes a new instance of an InterruptedIOException object with the given message.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InterruptedIOException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
InterruptedIOException Class
Concepts
InterruptedIOException Members
java.io Package

InterruptedIOException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an InterruptedIOException object during deserialization.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.InterruptedIOException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
InterruptedIOException Class
Concepts
InterruptedIOException Members
java.io Package

InterruptedIOException Constructor (String, Exception)

Initializes a new instance of an InterruptedIOException object with the given message and inner exception.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InterruptedIOException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the InterruptedIOException.

See Also

Reference

InterruptedIOException Class

Concepts

InterruptedIOException Members

java.io Package

InterruptedIOException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden. Retrieves the data to be serialized.
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InterruptedIOException Class

Concepts

java.io Package

InterruptedIOException.GetObjectData Method

Retrieves the data to be serialized.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization process.

See Also

Reference

InterruptedIOException Class

Concepts

InterruptedIOException Members

java.io Package

InterruptedIOException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

InterruptedIOException Class

Concepts

java.io Package

InvalidClassException Class

The exception that is thrown when an attempt is made to serialize or deserialize an instance of a class that does not support serialization.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.InvalidClassException
    extends java.io.ObjectStreamException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.ObjectStreamException

java.io.InvalidClassException

See Also

Concepts

InvalidClassException Members

java.io Package

InvalidClassException Members

The exception that is thrown when an attempt is made to serialize or deserialize an instance of a class that does not support serialization.

The following tables list the members exposed by the InvalidClassException type.

Public Constructors

Name	Description
InvalidClassException	Overloaded. Initializes a new instance of an InvalidClassException object.

Public Fields

Name	Description
classname	The name of the class that cannot be serialized or deserialized.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	Overridden. Provides the message describing the exceptional condition.
GetObjectData	Overridden. Retrieves the data to be serialized.
getClass	(inherited from Object)

clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InvalidClassException Class

Concepts

java.io Package

InvalidClassException Fields

Public Fields

Name	Description
classname	The name of the class that cannot be serialized or deserialized.

See Also

Reference

InvalidClassException Class

Concepts

java.io Package

InvalidClassException.classname Field

The name of the class that cannot be serialized or deserialized.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String classname;
```

See Also

Reference

InvalidClassException Class

Concepts

InvalidClassException Members

java.io Package

InvalidClassException Constructor

Initializes a new instance of an InvalidClassException object.

Overload List

Name	Description
InvalidClassException (String)	Initializes a new instance of an InvalidClassException object with the given message.
InvalidClassException (SerializationInfo, StreamingContext)	Initializes a new instance of an InvalidClassException object during deserialization.
InvalidClassException (String, Exception)	Initializes a new instance of an InvalidClassException object with the given message and inner exception.
InvalidClassException (String, String)	Initializes a new instance of an InvalidClassException object with the given class name and message.
InvalidClassException (String, String, Exception)	Initializes a new instance of an InvalidClassException object with the given class name, message, and inner exception.

See Also

Reference

InvalidClassException Class

Concepts

InvalidClassException Members

java.io Package

InvalidClassException Constructor (String)

Initializes a new instance of an InvalidClassException object with the given message.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InvalidClassException(
    java.lang.String reason);
```

Parameters

reason

A message describing the exceptional condition.

See Also
Reference
InvalidClassException Class
Concepts
InvalidClassException Members
java.io Package

InvalidClassException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an InvalidClassException object during deserialization.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.InvalidClassException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

InvalidClassException Class

Concepts

InvalidClassException Members

java.io Package

InvalidClassException Constructor (String, Exception)

Initializes a new instance of an InvalidClassException object with the given message and inner exception.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InvalidClassException(
    java.lang.String reason,
    System.Exception inner);
```

Parameters

reason

A message describing the exceptional condition.

inner

An inner exception that led to the InvalidClassException.

See Also
Reference
InvalidClassException Class
Concepts
InvalidClassException Members
java.io Package

InvalidClassException Constructor (String, String)

Initializes a new instance of an InvalidClassException object with the given class name and message.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InvalidClassException(
    java.lang.String className,
    java.lang.String reason);
```

Parameters

className

The name of the class that cannot be serialized or deserialized.

reason

A message describing the exceptional condition.

See Also

Reference

InvalidClassException Class

Concepts

InvalidClassException Members

java.io Package

InvalidClassException Constructor (String, String, Exception)

Initializes a new instance of an InvalidClassException object with the given class name, message, and inner exception.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InvalidClassException(
    java.lang.String className,
    java.lang.String reason,
    System.Exception inner);
```

Parameters

className

The name of the class that cannot be serialized or deserialized.

reason

A message describing the exceptional condition.

inner

An inner exception that led to the InvalidClassException.

See Also

Reference

InvalidClassException Class

Concepts

InvalidClassException Members

java.io Package

InvalidClassException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	Overridden. Provides the message describing the exceptional condition.
GetObjectData	Overridden. Retrieves the data to be serialized.
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InvalidClassException Class

Concepts

java.io Package

InvalidClassException.getMessage Method

Provides the message describing the exceptional condition.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getMessage();
```

Return Value

A message describing the exceptional condition.

See Also

Reference

InvalidClassException Class

Concepts

InvalidClassException Members

java.io Package

InvalidClassException.GetObjectData Method

Retrieves the data to be serialized.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization process.

See Also

Reference

InvalidClassException Class

Concepts

InvalidClassException Members

java.io Package

InvalidClassException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

InvalidClassException Class

Concepts

java.io Package

InvalidObjectException Class

The exception that is thrown when an error occurs during deserialization of an object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.InvalidObjectException
    extends java.io.ObjectStreamException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.ObjectStreamException

java.io.InvalidObjectException

See Also

Concepts

InvalidObjectException Members

java.io Package

InvalidObjectException Members

The exception that is thrown when an error occurs during deserialization of an object.

The following tables list the members exposed by the InvalidObjectException type.

Public Constructors

Name	Description
InvalidObjectException	Overloaded. Initializes a new instance of an InvalidObjectException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InvalidObjectException Class

Concepts

java.io Package

InvalidObjectException Constructor

Initializes a new instance of an InvalidObjectException object.

Overload List

Name	Description
InvalidObjectException (String)	Initializes a new instance of an InvalidObjectException object with the given message.
InvalidObjectException (SerializationInfo, StreamingContext)	Initializes a new instance of an InvalidObjectException object during deserialization.
InvalidObjectException (String, Exception)	Initializes a new instance of an InvalidObjectException object with the given message and inner exception.

See Also

Reference

InvalidObjectException Class

Concepts

InvalidObjectException Members

java.io Package

InvalidObjectException Constructor (String)

Initializes a new instance of an InvalidObjectException object with the given message.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InvalidObjectException(
    java.lang.String msg);
```

Parameters

msg

A message describing the exceptional condition.

See Also
Reference
InvalidObjectException Class
Concepts
InvalidObjectException Members
java.io Package

InvalidObjectException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an InvalidObjectException object during deserialization.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.InvalidObjectException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

InvalidObjectException Class

Concepts

InvalidObjectException Members

java.io Package

InvalidObjectException Constructor (String, Exception)

Initializes a new instance of an InvalidObjectException object with the given message and inner exception.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.InvalidObjectException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the InvalidObjectException.

See Also

Reference

InvalidObjectException Class

Concepts

InvalidObjectException Members

java.io Package

InvalidObjectException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InvalidObjectException Class

Concepts

java.io Package

InvalidObjectException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

InvalidObjectException Class

Concepts

java.io Package

IOException Class

The exception that is thrown when a generic I/O error has occurred. This class represents the base class for most of the exceptions in the java.io package.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.IOException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

Derived Classes

See Also

Concepts

IOException Members

java.io Package

IOException Members

The exception that is thrown when a generic I/O error has occurred. This class represents the base class for most of the exceptions in the java.io package.

The following tables list the members exposed by the IOException type.

Public Constructors

Name	Description
IOException	Overloaded. Initializes a new instance of an IOException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IOException Class

Concepts

java.io Package

IOException Constructor

Initializes a new instance of an IOException object.

Overload List

Name	Description
IOException ()	Initializes a new instance of an IOException object.
IOException (String)	Initializes a new instance of an IOException object with the given message.
IOException (SerializationInfo, StreamingContext)	Initializes a new instance of an IOException object during deserialization.
IOException (String, Exception)	Initializes a new instance of an IOException object with the given message and inner exception.

See Also

Reference

IOException Class

Concepts

IOException Members

java.io Package

IOException Constructor ()

Initializes a new instance of an IOException object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.IOException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

IOException Class

Concepts

IOException Members

java.io Package

IOException Constructor (String)

Initializes a new instance of an IOException object with the given message.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.IOException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
IOException Class
Concepts
IOException Members
java.io Package

IOException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an IOException object during deserialization.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.IOException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

IOException Class

Concepts

IOException Members

java.io Package

IOException Constructor (String, Exception)

Initializes a new instance of an IOException object with the given message and inner exception.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.IOException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the IOException.

See Also

Reference

IOException Class

Concepts

IOException Members

java.io Package

IOException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IOException Class

Concepts

java.io Package

IOException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

IOException Class

Concepts

java.io Package

LineNumberInputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.LineNumberInputStream
    extends java.io.FilterInputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

    java.io.FilterInputStream

      java.io.LineNumberInputStream

See Also

Concepts

LineNumberInputStream Members

java.io Package

LineNumberInputStream Members

The following tables list the members exposed by the LineNumberInputStream type.

Public Constructors

Name	Description
LineNumberInputStream	

Public Fields

Name	Description
in	(inherited from FilterInputStream)

Public Methods

Name	Description
available	Overridden.
close	Overridden.
equals	(inherited from Object)
hashCode	(inherited from Object)
getLineNumber	
getClass	(inherited from Object)
mark	Overridden.
markSupported	(inherited from FilterInputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden.
setLineNumber	
skip	Overridden.
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

LineNumberInputStream Class

Concepts

java.io Package

LineNumberInputStream Fields

Public Fields

Name	Description
in	(inherited from FilterInputStream)

See Also

Reference

LineNumberInputStream Class

Concepts

java.io Package

LineNumberInputStream Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.LineNumberInputStream(
    java.io.InputStream in);
```

Parameters

in

See Also

Reference

LineNumberInputStream Class

Concepts

LineNumberInputStream Members

java.io Package

LineNumberInputStream Methods

Public Methods

Name	Description
available	Overridden.
close	Overridden.
equals	(inherited from Object)
hashCode	(inherited from Object)
getLineNumber	
getClass	(inherited from Object)
mark	Overridden.
markSupported	(inherited from FilterInputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden.
setLineNumber	
skip	Overridden.
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

LineNumberInputStream Class

Concepts

java.io Package

LineNumberInputStream.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int available() throws java.io.IOException;
```

See Also

Reference

LineNumberInputStream Class

Concepts

LineNumberInputStream Members

java.io Package

LineNumberInputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

LineNumberInputStream Class

Concepts

LineNumberInputStream Members

java.io Package

LineNumberInputStream.getLineNumber Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int getLineNumber();
```

See Also

Reference

LineNumberInputStream Class

Concepts

LineNumberInputStream Members

java.io Package

LineNumberInputStream.mark Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void mark(
    int readlimit);
```

Parameters
readlimit

See Also
Reference
LineNumberInputStream Class
Concepts
LineNumberInputStream Members
java.io Package

LineNumberInputStream.read Method

Overload List

Name	Description
LineNumberInputStream.read ()	
LineNumberInputStream.read (byte[])	
LineNumberInputStream.read (byte[], int, int)	

See Also

Reference

LineNumberInputStream Class

Concepts

LineNumberInputStream Members

java.io Package

LineNumberInputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

See Also

Reference

LineNumberInputStream Class

Concepts

LineNumberInputStream Members

java.io Package

LineNumberInputStream.read Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

LineNumberInputStream Class

Concepts

LineNumberInputStream Members

java.io Package

LineNumberInputStream.reset Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void reset() throws java.io.IOException;
```

See Also

Reference

LineNumberInputStream Class

Concepts

LineNumberInputStream Members

java.io Package

LineNumberInputStream.setLineNumber Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void setLineNumber(
    int lineNumber);
```

Parameters
lineNumber

See Also
Reference
LineNumberInputStream Class
Concepts
LineNumberInputStream Members
java.io Package

LineNumberInputStream.skip Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long n) throws java.io.IOException;
```

Parameters

n

See Also

Reference

LineNumberInputStream Class

Concepts

LineNumberInputStream Members

java.io Package

LineNumberReader Class

Provides an implementation of the abstract Reader class that counts the number of lines in a stream of data.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.LineNumberReader
    extends java.io.BufferedReader
```

Example

The following example demonstrates the getLineNumber, mark, readLine, and reset methods of the LineNumberReader class.

```
// linenumberreader_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a LineNumberReader object
            // that is reading from a FileReader object.
            File myFile = new File("C:\\MyFile.txt");
            FileReader fileReader = new FileReader(myFile);
            LineNumberReader reader = new LineNumberReader(fileReader);

            // Read one line from the FileReader.
            String lineRead = "";
            lineRead = reader.readLine();
            System.out.println(lineRead);

            // Mark the position in the file.
            reader.mark((int)myFile.length());

            // Read the rest of the file.
            while ((lineRead = reader.readLine()) != null)
            {
                System.out.println(lineRead);
            }

            // Print out the number of lines that were read.
            System.out.println("Total lines read: " +
                reader.getLineNumber());

            // Now reset the reader and re-read from the
            // FileReader from the marked position on.
            reader.reset();
            while ((lineRead = reader.readLine()) != null)
            {
                System.out.println(lineRead);
            }

            // Close the LineNumberReader and FileReader.
            fileReader.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
```

```
            }
        }
    }

    /*
    Output:
    This is my file.
    It contains three lines.
    This is the last line.
    Total lines read: 3

    It contains three lines.
    This is the last line.
    */
```

Remarks

This class extends from BufferedReader and encapsulates a Reader object as the source of the stream. This provides functionality for counting the number of lines in the stream and allows for mark and reset functionality on the number of lines read from the stream of character data.

Inheritance Hierarchy

java.lang.Object

  java.io.Reader

    java.io.BufferedReader

      java.io.LineNumberReader

See Also

Concepts

LineNumberReader Members

java.io Package

LineNumberReader Members

Provides an implementation of the abstract Reader class that counts the number of lines in a stream of data.

The following tables list the members exposed by the LineNumberReader type.

Public Constructors

Name	Description
LineNumberReader	Overloaded. Initializes a new instance of a LineNumberReader object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader.(inherited from Reader)

Public Methods

Name	Description
close	Closes the stream being buffered and releases internal buffer. (inherited from BufferedReader)
equals	(inherited from Object)
hashCode	(inherited from Object)
getLineNumber	Gets the current number of lines read from the input source.
getClass	(inherited from Object)
mark	Overridden. Sets the mark to the current line being read from the stream. Also specifies the maximum numbe r of characters that can be marked.
markSupported	Determines whether the current position of the stream can be marked. (inherited from BufferedReader)
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the input source.
readLine	Overridden. Reads an entire line from the input source.
ready	Determines whether the underlying stream is ready for reading. (inherited from BufferedReader)
reset	Overridden. Resets the reader for the input source such that the next character read is the character that is ma rked.
setLineNumber	Sets the number of lines read from the input source.
skip	Overridden. Skips over the next n characters in the input source or until the end of the stream, whichever is les s.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description

finalize	(inherited from Object)

See Also

Reference

LineNumberReader Class

Concepts

java.io Package

LineNumberReader Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader. (inherited from Reader)

See Also

Reference

LineNumberReader Class

Concepts

java.io Package

LineNumberReader Constructor

Initializes a new instance of a LineNumberReader object.

Overload List

Name	Description
LineNumberReader (Reader)	Initializes a new instance of a LineNumberReader object. The provided Reader object contains the data whose lines will be counted.
LineNumberReader (Reader, int)	Initializes a new instance of a LineNumberReader object. The provided Reader object contains the data whose lines will be counted. The internal buffer is initialized to the given size.

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

LineNumberReader Constructor (Reader)

Initializes a new instance of a LineNumberReader object. The provided Reader object contains the data whose lines will be counted.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.LineNumberReader(
    java.io.Reader in);
```

Parameters

in

A Reader object that contains the data whose lines will be counted.

See Also
Reference
LineNumberReader Class
Concepts
LineNumberReader Members
java.io Package

LineNumberReader Constructor (Reader, Int32)

Initializes a new instance of a LineNumberReader object. The provided Reader object contains the data whose lines will be counted. The internal buffer is initialized to the given size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.LineNumberReader(
    java.io.Reader in,
    int size);
```

Parameters

in

A Reader object that contains the data whose lines will be counted.

size

The size of the internal buffer.

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

LineNumberReader Methods

Public Methods

Name	Description
close	Closes the stream being buffered and releases internal buffer. (inherited from BufferedReader)
equals	(inherited from Object)
hashCode	(inherited from Object)
getLineNumber	Gets the current number of lines read from the input source.
getClass	(inherited from Object)
mark	Overridden. Sets the mark to the current line being read from the stream. Also specifies the maximum numbe r of characters that can be marked.
markSupported	Determines whether the current position of the stream can be marked. (inherited from BufferedReader)
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the input source.
readLine	Overridden. Reads an entire line from the input source.
ready	Determines whether the underlying stream is ready for reading. (inherited from BufferedReader)
reset	Overridden. Resets the reader for the input source such that the next character read is the character that is ma rked.
setLineNumber	Sets the number of lines read from the input source.
skip	Overridden. Skips over the next n characters in the input source or until the end of the stream, whichever is les s.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

LineNumberReader Class

Concepts

java.io Package

LineNumberReader.getLineNumber Method

Gets the current number of lines read from the input source.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int getLineNumber();
```

Return Value

The current number of lines read from the input source.

Example

The following example demonstrates how to determine the number of lines read using a LineNumberReader object.

```
// linenumberreader_getlinenumber.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a LineNumberReader object
            // that is reading from a FileReader object.
            File myFile = new File("C:\\MyFile.txt");
            FileReader fileReader = new FileReader(myFile);
            LineNumberReader reader = new LineNumberReader(fileReader);

            // Read from the FileReader.
            String lineRead = "";
            while ((lineRead = reader.readLine()) != null)
            {
                System.out.println(lineRead);
            }

            // Determine the number of lines that were read.
            System.out.println("Total lines read: " +
                reader.getLineNumber());

            // Close the LineNumberReader and FileReader.
            fileReader.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is my file.
It contains three lines.
This is the last line.
Total lines read: 3
*/
```

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

LineNumberReader.mark Method

Sets the mark to the current line being read from the stream. Also specifies the maximum number of characters that can be marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void mark(
    int readlimit) throws java.io.IOException;
```

Parameters

readlimit

The maximum number of characters that can be marked.

Example

The following example demonstrates the effects that calling mark and reset have on reading data using a LineNumberReader object.

```
// linenumberreader_mark.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a LineNumberReader object
            // that is reading from a FileReader object.
            File myFile = new File("C:\\MyFile.txt");
            FileReader fileReader = new FileReader(myFile);
            LineNumberReader reader = new LineNumberReader(fileReader);

            // Read one line from the FileReader.
            String lineRead = "";
            lineRead = reader.readLine();
            System.out.println(lineRead);

            // Mark the position in the file.
            reader.mark((int)myFile.length());

            // Read the rest of the file.
            while ((lineRead = reader.readLine()) != null)
            {
                System.out.println(lineRead);
            }

            // Now reset the reader and re-read from the
            // FileReader from the marked position on.
            reader.reset();
            while ((lineRead = reader.readLine()) != null)
            {
                System.out.println(lineRead);
            }

            // Close the LineNumberReader and FileReader.
            fileReader.close();
            reader.close();
        }
```

```
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is my file.
It contains three lines.
This is the last line.

It contains three lines.
This is the last line.
*/
```

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

LineNumberReader.read Method

Reads the next character or characters from the input source.

Overload List

Name	Description
LineNumberReader.read ()	Reads the next character from the input source.
LineNumberReader.read (char[])	Reads the characters in the array starting at an offset for the length specified and stores them in the provided array of characters.
LineNumberReader.read (char[], int, int)	Reads the characters from the underlying stream for the length specified, and stores them in the provided array of characters, starting at an offset.

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

LineNumberReader.read Method ()

Reads the next character from the input source.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

Return Value

The numeric value of the character read from the stream, or -1 if the end of the stream is reached. A line feed (LF) character is returned if a new line is detected.

Note:
CR+LF is treated as a new line.

Example

The following example demonstrates how to read a single character at a time using a LineNumberReader object.

```
// linenumberreader_read.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a LineNumberReader object
            // that is reading from a FileReader object.
            FileReader fileReader = new FileReader("C:\\MyFile.txt");
            LineNumberReader reader = new LineNumberReader(fileReader);

            // Read from the FileReader.
            int charRead;
            while ((charRead = reader.read()) >= 0)
            {
                System.out.print((char)charRead);
            }

            // Close the LineNumberReader and FileReader.
            fileReader.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is my file.
It contains three lines.
This is the last line.
*/
```

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

LineNumberReader.read Method (Char[ ], Int32, Int32)

Reads the characters from the underlying stream for the length specified, and stores them in the provided array of characters, starting at an offset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

The array of characters to store the data read from the input source.

off

An offset into the array of characters. This value represents the index of the first character in the array to be written. This value must be greater than zero.

len

The number of characters to be read. This value plus the offset must be less than the overall length of the array.

Return Value

The number of characters read from the stream, or -1 if the end of the stream is reached and no characters are read.

Example

The following example demonstrates how to store the data read from a LineNumberReader object into an array.

```
// linenumberreader_read_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a LineNumberReader object
            // that is reading from a FileReader object.
            File myFile = new File("C:\\MyFile.txt");
            FileReader fileReader = new FileReader(myFile);
            LineNumberReader reader = new LineNumberReader(fileReader);

            // Read from the FileReader.
            char[] arr = new char[(int)myFile.length()];
            reader.read(arr, 0, (int)myFile.length());
            System.out.println(arr);

            // Close the LineNumberReader and FileReader.
            fileReader.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
```

```
}

/*
Output:
This is my file.
It contains three lines.
This is the last line.
*/
```

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

LineNumberReader.readLine Method

Reads an entire line from the input source.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String readLine() throws java.io.IOException;
```

Return Value

A string representing the contents of the line read from the underlying stream. The string contains all the text in the stream up to the next carriage return (CR), line feed (LF), carriage return plus line feed (CR+LF), or the end of the stream. If there are no characters other than CR or LF in the stream, then null is returned.

Example

The following example demonstrates how to read a line of data using a LineNumberReader object.

```
// linenumberreader_readline.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a LineNumberReader object
            // that is reading from a FileReader object.
            File myFile = new File("C:\\MyFile.txt");
            FileReader fileReader = new FileReader(myFile);
            LineNumberReader reader = new LineNumberReader(fileReader);

            // Read from the FileReader.
            String lineRead = "";
            while ((lineRead = reader.readLine()) != null)
            {
                System.out.println(lineRead);
            }

            // Close the LineNumberReader and FileReader.
            fileReader.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is my file.
It contains three lines.
This is the last line.
*/
```

See Also

Reference

LineNumberReader Class

Concepts
LineNumberReader Members
java.io Package

LineNumberReader.reset Method

Resets the reader for the input source such that the next character read is the character that is marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void reset() throws java.io.IOException;
```

Example

The following example demonstrates the effects that calling mark and reset have on reading data using a LineNumberReader object.

```
// linenumberreader_reset.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a LineNumberReader object
            // that is reading from a FileReader object.
            File myFile = new File("C:\\MyFile.txt");
            FileReader fileReader = new FileReader(myFile);
            LineNumberReader reader = new LineNumberReader(fileReader);

            // Read one line from the FileReader.
            String lineRead = "";
            lineRead = reader.readLine();
            System.out.println(lineRead);

            // Mark the position in the file.
            reader.mark((int)myFile.length());

            // Read the rest of the file.
            while ((lineRead = reader.readLine()) != null)
            {
                System.out.println(lineRead);
            }

            // Now reset the reader and re-read from the
            // FileReader from the marked position on.
            reader.reset();
            while ((lineRead = reader.readLine()) != null)
            {
                System.out.println(lineRead);
            }

            // Close the LineNumberReader and FileReader.
            fileReader.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
This is my file.
It contains three lines.
This is the last line.

It contains three lines.
This is the last line.
*/
```

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

LineNumberReader.setLineNumber Method

Sets the number of lines read from the input source.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void setLineNumber(
    int lineNumber);
```

Parameters

lineNumber

A value representing the number of lines read from the input source.

Example

The following example demonstrates how to set the number of lines read using a LineNumberReader object.

```
// linenumberreader_setlinenumber.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a LineNumberReader object
            // that is reading from a FileReader object.
            File myFile = new File("C:\\MyFile.txt");
            FileReader fileReader = new FileReader(myFile);
            LineNumberReader reader = new LineNumberReader(fileReader);

            // Read from the FileReader.
            String lineRead = "";
            while ((lineRead = reader.readLine()) != null)
            {
                System.out.println(lineRead);
            }

            // Determine the number of lines that were read.
            System.out.println("Total lines read: " +
                reader.getLineNumber());

            // Reset the number of lines read.
            reader.setLineNumber(0);
            System.out.println("Total lines read after reset: " +
                reader.getLineNumber());

            // Close the LineNumberReader and FileReader.
            fileReader.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
```

```
This is my file.
It contains three lines.
This is the last line.
Total lines read: 3
Total lines read after reset: 0
*/
```

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

LineNumberReader.skip Method

Skips over the next n characters in the input source or until the end of the stream, whichever is less.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long n) throws java.io.IOException;
```

Parameters

n

The number of characters to skip over. This value is treated as zero if it is negative. If this number is greater than the number of characters remaining in the stream, then only the number of remaining characters will be skipped.

Return Value

The number of characters skipped.

Example

The following example shows how to skip over data being read and the effect it has on the count of the number of lines read.

```
// linenumberreader_skip.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a LineNumberReader object
            // that is reading from a FileReader object.
            File myFile = new File("C:\\MyFile.txt");
            FileReader fileReader = new FileReader(myFile);
            LineNumberReader reader = new LineNumberReader(fileReader);

            // Read from the FileReader.
            int charRead;
            int skipCount = 0;
            while ((charRead = reader.read()) >= 0)
            {
                System.out.print((char)charRead);

                // Skip over some characters.
                reader.skip(++skipCount);
            }

            // Determine the number of lines that were read.
            System.out.println("\nTotal lines read: " +
                reader.getLineNumber());

            // Close the LineNumberReader and FileReader.
            fileReader.close();
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
```

```
}

/*
Output:
Tiiyecslil
Total lines read: 2
*/
```

See Also

Reference

LineNumberReader Class

Concepts

LineNumberReader Members

java.io Package

NotActiveException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.NotActiveException
    extends java.io.ObjectStreamException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.ObjectStreamException

java.io.NotActiveException

See Also

Concepts

NotActiveException Members

java.io Package

NotActiveException Members

The following tables list the members exposed by the NotActiveException type.

Public Constructors

Name	Description
NotActiveException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NotActiveException Class

Concepts

java.io Package

NotActiveException Constructor

Overload List

Name	Description
NotActiveException ()	
NotActiveException (String)	
NotActiveException (SerializationInfo, StreamingContext)	
NotActiveException (String, Exception)	

See Also

Reference

NotActiveException Class

Concepts

NotActiveException Members

java.io Package

NotActiveException Constructor ()

Initializes a new instance of the NotActiveException Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.NotActiveException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NotActiveException Class

Concepts

NotActiveException Members

java.io Package

NotActiveException Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.NotActiveException(
    java.lang.String msg);
```

Parameters

msg

See Also
Reference
NotActiveException Class
Concepts
NotActiveException Members
java.io Package

NotActiveException Constructor (SerializationInfo, StreamingContext)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.NotActiveException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

NotActiveException Class

Concepts

NotActiveException Members

java.io Package

NotActiveException Constructor (String, Exception)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.NotActiveException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also

Reference

NotActiveException Class

Concepts

NotActiveException Members

java.io Package

NotActiveException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NotActiveException Class

Concepts

java.io Package

NotActiveException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NotActiveException Class

Concepts

java.io Package

NotSerializableException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.NotSerializableException
    extends java.io.ObjectStreamException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.ObjectStreamException

java.io.NotSerializableException

See Also

Concepts

NotSerializableException Members

java.io Package

NotSerializableException Members

The following tables list the members exposed by the NotSerializableException type.

Public Constructors

Name	Description
NotSerializableException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NotSerializableException Class

Concepts

java.io Package

NotSerializableException Constructor

Overload List

Name	Description
NotSerializableException ()	
NotSerializableException (String)	
NotSerializableException (SerializationInfo, StreamingContext)	
NotSerializableException (String, Exception)	

See Also

Reference

NotSerializableException Class

Concepts

NotSerializableException Members

java.io Package

NotSerializableException Constructor ()

Initializes a new instance of the NotSerializableException Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.NotSerializableException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NotSerializableException Class

Concepts

NotSerializableException Members

java.io Package

NotSerializableException Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.NotSerializableException(
    java.lang.String msg);
```

Parameters

msg

See Also

Reference

NotSerializableException Class

Concepts

NotSerializableException Members

java.io Package

NotSerializableException Constructor (SerializationInfo, StreamingContext)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.NotSerializableException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

NotSerializableException Class

Concepts

NotSerializableException Members

java.io Package

NotSerializableException Constructor (String, Exception)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.NotSerializableException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also

Reference

NotSerializableException Class

Concepts

NotSerializableException Members

java.io Package

NotSerializableException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NotSerializableException Class

Concepts

java.io Package

NotSerializableException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NotSerializableException Class

Concepts

java.io Package

ObjectInput Interface

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public interface java.io.ObjectInput
    extends java.io.DataInput
```

See Also

Concepts

ObjectInput Members

java.io Package

ObjectInput Members

The following tables list the members exposed by the ObjectInput type.

Public Methods

Name	Description
available	
close	
read	Overloaded.
readObject	
skip	

See Also

Reference

ObjectInput Interface

Concepts

java.io Package

ObjectInput Methods

Public Methods

Name	Description
available	
close	
read	Overloaded.
readObject	
skip	

See Also

Reference

ObjectInput Interface

Concepts

java.io Package

ObjectInput.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int available() throws java.io.IOException;
```

See Also

Reference

ObjectInput Interface

Concepts

ObjectInput Members

java.io Package

ObjectInput.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void close() throws java.io.IOException;
```

See Also

Reference

ObjectInput Interface

Concepts

ObjectInput Members

java.io Package

ObjectInput.read Method

Overload List

Name	Description
ObjectInput.read ()	
ObjectInput.read (byte[])	
ObjectInput.read (byte[], int, int)	

See Also

Reference

ObjectInput Interface

Concepts

ObjectInput Members

java.io Package

ObjectInput.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int read() throws java.io.IOException;
```

See Also

Reference

ObjectInput Interface

Concepts

ObjectInput Members

java.io Package

ObjectInput.read Method (SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int read(
    byte[] buf) throws java.io.IOException;
```

Parameters

buf

See Also
Reference
ObjectInput Interface
Concepts
ObjectInput Members
java.io Package

ObjectInput.read Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int read(
    byte[] buf,
    int off,
    int count) throws java.io.IOException;
```

Parameters

buf

off

count

See Also

Reference

ObjectInput Interface

Concepts

ObjectInput Members

java.io Package

ObjectInput.readObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object readObject() throws java.lang.ClassNotFoundException, java
.io.IOException;
```

See Also

Reference

ObjectInput Interface

Concepts

ObjectInput Members

java.io Package

ObjectInput.skip Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract long skip(
    long count) throws java.io.IOException;
```

Parameters

count

See Also
Reference
ObjectInput Interface
Concepts
ObjectInput Members
java.io Package

ObjectInputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.ObjectInputStream
    extends java.io.InputStream
    implements java.io.ObjectInput, java.io.ObjectStreamConstants
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

   java.io.ObjectInputStream

See Also

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream Members

The following tables list the members exposed by the ObjectInputStream type.

Public Constructors

Name	Description
ObjectInputStream	

Public Methods

Name	Description
available	Overridden.
close	Overridden.
defaultReadObject	
enableResolveObject	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
readBoolean	
readByte	
readChar	
readDouble	
readFloat	
readFully	Overloaded.
readInt	
readLine	
readLong	
readObject	

readShort	
readStreamHeader	
readUnsignedByte	
readUnsignedShort	
readUTF	
registerValidation	
reset	(inherited from InputStream)
resolveClass	
resolveObject	
skip	(inherited from InputStream)
skipBytes	
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ObjectInputStream Class

Concepts

java.io Package

ObjectInputStream Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.ObjectInputStream(
    java.io.InputStream in) throws java.io.IOException, java.io.StreamCorruptedException;
```

Parameters

in

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream Methods

Public Methods

Name	Description
available	Overridden.
close	Overridden.
defaultReadObject	
enableResolveObject	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
readBoolean	
readByte	
readChar	
readDouble	
readFloat	
readFully	Overloaded.
readInt	
readLine	
readLong	
readObject	
readShort	
readStreamHeader	
readUnsignedByte	

readUnsignedShort	
readUTF	
registerValidation	
reset	(inherited from InputStream)
resolveClass	
resolveObject	
skip	(inherited from InputStream)
skipBytes	
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ObjectInputStream Class

Concepts

java.io Package

ObjectInputStream.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int available() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.defaultReadObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void defaultReadObject() throws java.io.IOException, java.lang.ClassNotFoundEx
ception, java.io.NotActiveException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.enableResolveObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected final boolean enableResolveObject(
    boolean enable) throws java.lang.SecurityException;
```

Parameters

enable

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.read Method

Overload List

Name	Description
ObjectInputStream.read ()	
ObjectInputStream.read (byte[])	
ObjectInputStream.read (byte[], int, int)	

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.read Method (SByte[])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] buffer) throws java.io.IOException;
```

Parameters

buffer

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.read Method (SByte[], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] buffer,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buffer

offset

count

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readBoolean Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean readBoolean() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readByte Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public byte readByte() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readChar Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public char readChar() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readDouble Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public double readDouble() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readFloat Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public float readFloat() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readFully Method

Overload List

Name	Description
ObjectInputStream.readFully (byte[])	
ObjectInputStream.readFully (byte[], int, int)	

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readFully Method ((SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void readFully(
    byte[] buffer) throws java.io.IOException;
```

Parameters

buffer

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readFully Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void readFully(
    byte[] buffer,
    int offset,
    int len) throws java.io.IOException;
```

Parameters

buffer

offset

len

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readInt Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int readInt() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readLine Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String readLine() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readLong Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long readLong() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.Object readObject() throws java.io.OptionalDataException, java.lang.
ClassNotFoundException, java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readShort Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public short readShort() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readStreamHeader Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void readStreamHeader() throws java.io.IOException, java.io.StreamCorruptedExcept
ion;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readUnsignedByte Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int readUnsignedByte() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readUnsignedShort Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int readUnsignedShort() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.readUTF Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String readUTF() throws java.io.IOException;
```

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.registerValidation Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void registerValidation(
    java.io.ObjectInputValidation obj,
    int prio) throws java.io.NotActiveException, java.io.InvalidObjectException;
```

Parameters

obj

prio

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.resolveClass Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Class resolveClass(
    java.io.ObjectStreamClass desc) throws java.io.IOException, java.lang.ClassNotFoundExce
ption;
```

Parameters

desc

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.resolveObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Object resolveObject(
    java.lang.Object obj) throws java.io.IOException;
```

Parameters

obj

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputStream.skipBytes Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int skipBytes(
    int count) throws java.io.IOException;
```

Parameters

count

See Also

Reference

ObjectInputStream Class

Concepts

ObjectInputStream Members

java.io Package

ObjectInputValidation Interface

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public interface java.io.ObjectInputValidation
```

See Also

Concepts

ObjectInputValidation Members

java.io Package

ObjectInputValidation Members

The following tables list the members exposed by the ObjectInputValidation type.

Public Methods

Name	Description
validateObject	

See Also

Reference

ObjectInputValidation Interface

Concepts

java.io Package

ObjectInputValidation Methods

Public Methods

Name	Description
validateObject	

See Also

Reference

ObjectInputValidation Interface

Concepts

java.io Package

ObjectInputValidation.validateObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void validateObject() throws java.io.InvalidObjectException;
```

See Also

Reference

ObjectInputValidation Interface

Concepts

ObjectInputValidation Members

java.io Package

ObjectOutput Interface

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public interface java.io.ObjectOutput
    extends java.io.DataOutput
```

See Also

Concepts

ObjectOutput Members

java.io Package

ObjectOutput Members

The following tables list the members exposed by the ObjectOutput type.

Public Methods

Name	Description
close	
flush	
write	Overloaded.
writeObject	

See Also

Reference

ObjectOutput Interface

Concepts

java.io Package

ObjectOutput Methods

Public Methods

Name	Description
close	
flush	
write	Overloaded.
writeObject	

See Also

Reference

ObjectOutput Interface

Concepts

java.io Package

ObjectOutput.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void close() throws java.io.IOException;
```

See Also

Reference

ObjectOutput Interface

Concepts

ObjectOutput Members

java.io Package

ObjectOutput.flush Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void flush() throws java.io.IOException;
```

See Also

Reference

ObjectOutput Interface

Concepts

ObjectOutput Members

java.io Package

ObjectOutput.write Method

Overload List

Name	Description
ObjectOutput.write (int)	
ObjectOutput.write (byte[])	
ObjectOutput.write (byte[], int, int)	

See Also

Reference

ObjectOutput Interface

Concepts

ObjectOutput Members

java.io Package

ObjectOutput.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void write(
    int oneByte) throws java.io.IOException;
```

Parameters

oneByte

See Also

Reference

ObjectOutput Interface

Concepts

ObjectOutput Members

java.io Package

ObjectOutput.write Method (SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void write(
    byte[] buffer) throws java.io.IOException;
```

Parameters

buffer

See Also

Reference

ObjectOutput Interface

Concepts

ObjectOutput Members

java.io Package

ObjectOutput.write Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void write(
    byte[] buffer,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buffer

offset

count

See Also

Reference

ObjectOutput Interface

Concepts

ObjectOutput Members

java.io Package

ObjectOutput.writeObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void writeObject(
    java.lang.Object obj) throws java.io.IOException;
```

Parameters

obj

See Also

Reference

ObjectOutput Interface

Concepts

ObjectOutput Members

java.io Package

ObjectOutputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.ObjectOutputStream
    extends java.io.OutputStream
    implements java.io.ObjectOutput, java.io.ObjectStreamConstants
```

Inheritance Hierarchy

java.lang.Object

  java.io.OutputStream

  java.io.ObjectOutputStream

See Also

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream Members

The following tables list the members exposed by the ObjectOutputStream type.

Public Constructors

Name	Description
ObjectOutputStream	

Public Methods

Name	Description
annotateClass	
close	Overridden.
defaultWriteObject	
drain	
enableReplaceObject	
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
replaceObject	
reset	
toString	(inherited from OutputStream)
write	Overloaded. Overridden.
writeBoolean	
writeByte	
writeBytes	
writeChar	
writeChars	
writeDouble	
writeFloat	

writeInt	
writeLong	
writeObject	
writeShort	
writeStreamHeader	
writeUTF	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ObjectOutputStream Class

Concepts

java.io Package

ObjectOutputStream Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.ObjectOutputStream(
    java.io.OutputStream out) throws java.io.IOException;
```

Parameters

out

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream Methods

Public Methods

Name	Description
annotateClass	
close	Overridden.
defaultWriteObject	
drain	
enableReplaceObject	
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
replaceObject	
reset	
toString	(inherited from OutputStream)
write	Overloaded. Overridden.
writeBoolean	
writeByte	
writeBytes	
writeChar	
writeChars	
writeDouble	
writeFloat	
writeInt	
writeLong	
writeObject	

writeShort	
writeStreamHeader	
writeUTF	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ObjectOutputStream Class

Concepts

java.io Package

ObjectOutputStream.annotateClass Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void annotateClass(
    java.lang.Class cl) throws java.io.IOException;
```

Parameters

cl

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.defaultWriteObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void defaultWriteObject() throws java.io.IOException;
```

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.drain Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void drain() throws java.io.IOException;
```

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.enableReplaceObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected final boolean enableReplaceObject(
    boolean enable) throws java.lang.SecurityException;
```

Parameters

enable

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.flush Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush() throws java.io.IOException;
```

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.replaceObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Object replaceObject(
    java.lang.Object obj) throws java.io.IOException;
```

Parameters

obj

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.reset Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void reset() throws java.io.IOException;
```

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.write Method

Overload List

Name	Description
ObjectOutputStream.write (int)	
ObjectOutputStream.write (byte[])	
ObjectOutputStream.write (byte[], int, int)	

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int oneByte) throws java.io.IOException;
```

Parameters

oneByte

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.write Method (SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] buffer) throws java.io.IOException;
```

Parameters
buffer

See Also
Reference
ObjectOutputStream Class
Concepts
ObjectOutputStream Members
java.io Package

ObjectOutputStream.write Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] buffer,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buffer

offset

count

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeBoolean Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeBoolean(
    boolean val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeByte Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeByte(
    int val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeBytes Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeBytes(
    java.lang.String str) throws java.io.IOException;
```

Parameters

str

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeChar Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeChar(
    int val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeChars Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeChars(
    java.lang.String str) throws java.io.IOException;
```

Parameters

str

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeDouble Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeDouble(
    double val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeFloat Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeFloat(
    float val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeInt Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeInt(
    int val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeLong Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeLong(
    long val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeObject Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeObject(
    java.lang.Object obj) throws java.io.IOException;
```

Parameters

obj

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeShort Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeShort(
    int val) throws java.io.IOException;
```

Parameters

val

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeStreamHeader Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void writeStreamHeader() throws java.io.IOException;
```

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectOutputStream.writeUTF Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void writeUTF(
    java.lang.String str) throws java.io.IOException;
```

Parameters

str

See Also

Reference

ObjectOutputStream Class

Concepts

ObjectOutputStream Members

java.io Package

ObjectStreamClass Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.ObjectStreamClass
    extends java.lang.Object
    implements java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

java.io.ObjectStreamClass

See Also

Concepts

ObjectStreamClass Members

java.io Package

ObjectStreamClass Members

The following tables list the members exposed by the ObjectStreamClass type.

Public Methods

Name	Description
equals	(inherited from Object)
forClass	
hashCode	(inherited from Object)
getName	
getSerialVersionUID	
getClass	(inherited from Object)
lookup	
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ObjectStreamClass Class

Concepts

java.io Package

ObjectStreamClass Methods

Public Methods

Name	Description
equals	(inherited from Object)
forClass	
hashCode	(inherited from Object)
getName	
getSerialVersionUID	
getClass	(inherited from Object)
lookup	
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ObjectStreamClass Class

Concepts

java.io Package

ObjectStreamClass.clone Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

See Also

Reference

ObjectStreamClass Class

Concepts

ObjectStreamClass Members

java.io Package

ObjectStreamClass.forClass Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class forClass();
```

See Also

Reference

ObjectStreamClass Class

Concepts

ObjectStreamClass Members

java.io Package

ObjectStreamClass.getName Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getName();
```

See Also

Reference

ObjectStreamClass Class

Concepts

ObjectStreamClass Members

java.io Package

ObjectStreamClass.getSerialVersionUID Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long getSerialVersionUID();
```

See Also

Reference

ObjectStreamClass Class

Concepts

ObjectStreamClass Members

java.io Package

ObjectStreamClass.lookup Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static java.io.ObjectStreamClass lookup(
    java.lang.Class cl);
```

Parameters

cl

See Also

Reference

ObjectStreamClass Class

Concepts

ObjectStreamClass Members

java.io Package

ObjectStreamClass.toString Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

ObjectStreamClass Class

Concepts

ObjectStreamClass Members

java.io Package

ObjectStreamException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.io.ObjectStreamException
    extends java.io.IOException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.ObjectStreamException

Derived Classes

See Also

Concepts

ObjectStreamException Members

java.io Package

ObjectStreamException Members

The following tables list the members exposed by the ObjectStreamException type.

Public Constructors

Name	Description
ObjectStreamException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ObjectStreamException Class

Concepts

java.io Package

ObjectStreamException Constructor

Overload List

Name	Description
ObjectStreamException ()	
ObjectStreamException (String)	
ObjectStreamException (SerializationInfo, StreamingContext)	
ObjectStreamException (String, Exception)	

See Also

Reference

ObjectStreamException Class

Concepts

ObjectStreamException Members

java.io Package

ObjectStreamException Constructor ()

Initializes a new instance of the ObjectStreamException Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.ObjectStreamException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ObjectStreamException Class

Concepts

ObjectStreamException Members

java.io Package

ObjectStreamException Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.ObjectStreamException(
    java.lang.String s);
```

Parameters

s

See Also

Reference

ObjectStreamException Class

Concepts

ObjectStreamException Members

java.io Package

ObjectStreamException Constructor (SerializationInfo, StreamingContext)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.ObjectStreamException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

ObjectStreamException Class

Concepts

ObjectStreamException Members

java.io Package

ObjectStreamException Constructor (String, Exception)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.ObjectStreamException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also

Reference

ObjectStreamException Class

Concepts

ObjectStreamException Members

java.io Package

ObjectStreamException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ObjectStreamException Class

Concepts

java.io Package

ObjectStreamException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ObjectStreamException Class

Concepts

java.io Package

OptionalDataException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.OptionalDataException
    extends java.io.ObjectStreamException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.ObjectStreamException

java.io.OptionalDataException

See Also

Concepts

OptionalDataException Members

java.io Package

OptionalDataException Members

The following tables list the members exposed by the OptionalDataException type.

Public Constructors

Name	Description
OptionalDataException	

Public Fields

Name	Description
eof	
length	

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)

clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

OptionalDataException Class

Concepts

java.io Package

OptionalDataException Fields

Public Fields

Name	Description
eof	
length	

See Also

Reference

OptionalDataException Class

Concepts

java.io Package

OptionalDataException.eof Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean eof;
```

See Also

Reference

OptionalDataException Class

Concepts

OptionalDataException Members

java.io Package

OptionalDataException.length Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int length;
```

See Also

Reference

OptionalDataException Class

Concepts

OptionalDataException Members

java.io Package

OptionalDataException Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.OptionalDataException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also
Reference
OptionalDataException Class
Concepts
OptionalDataException Members
java.io Package

OptionalDataException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

OptionalDataException Class

Concepts

java.io Package

OptionalDataException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

OptionalDataException Class

Concepts

java.io Package

OutputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.io.OutputStream
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.io.OutputStream

java.io.ByteArrayOutputStream

java.io.FileOutputStream

java.io.FilterOutputStream

java.io.ObjectOutputStream

java.io.PipedOutputStream

See Also

Concepts

OutputStream Members

java.io Package

OutputStream Members

The following tables list the members exposed by the OutputStream type.

Public Constructors

Name	Description
OutputStream	

Public Methods

Name	Description
close	
equals	(inherited from Object)
flush	
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
toString	Overridden.
write	Overloaded.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

OutputStream Class

Concepts

java.io Package

OutputStream Constructor

Initializes a new instance of the OutputStream Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.OutputStream();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

OutputStream Class

Concepts

OutputStream Members

java.io Package

OutputStream Methods

Public Methods

Name	Description
close	
equals	(inherited from Object)
flush	
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
toString	Overridden.
write	Overloaded.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

OutputStream Class

Concepts

java.io Package

OutputStream.clone Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

See Also

Reference

OutputStream Class

Concepts

OutputStream Members

java.io Package

OutputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

OutputStream Class

Concepts

OutputStream Members

java.io Package

OutputStream.flush Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush() throws java.io.IOException;
```

See Also

Reference

OutputStream Class

Concepts

OutputStream Members

java.io Package

OutputStream.toString Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

See Also

Reference

OutputStream Class

Concepts

OutputStream Members

java.io Package

OutputStream.write Method

Overload List

Name	Description
OutputStream.write (int)	
OutputStream.write (byte[])	
OutputStream.write (byte[], int, int)	

See Also

Reference

OutputStream Class

Concepts

OutputStream Members

java.io Package

OutputStream.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void write(
    int b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

OutputStream Class

Concepts

OutputStream Members

java.io Package

OutputStream.write Method (SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

OutputStream Class

Concepts

OutputStream Members

java.io Package

OutputStream.write Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] buffer,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buffer

offset

count

See Also

Reference

OutputStream Class

Concepts

OutputStream Members

java.io Package

OutputStreamWriter Class

Provides an implementation of the abstract Writer class specific to writing characters to an OutputStream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.OutputStreamWriter
    extends java.io.Writer
```

Example

The following example demonstrates the close, flush, getEncoding, and write methods of the OutputStreamWriter class.

```
// outputstreamwriter_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a OutputStreamWriter object
            // attached to a ByteArrayOutputStream.
            ByteArrayOutputStream out =
                new ByteArrayOutputStream();
            OutputStreamWriter writer = new OutputStreamWriter(out);

            // Write to the output stream.
            String s = "Random bytes";
            writer.write(s);
            writer.flush();

            // Display the contents of the ByteArrayOutputStream.
            System.out.println(out.toString());

            // Display the encoding being used.
            System.out.println("encoding: " +
                writer.getEncoding());

            // Close the OutputStreamWriter object.
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Random bytes
encoding: Cp1252
*/
```

Inheritance Hierarchy

java.lang.Object

  java.io.Writer

java.io.OutputStreamWriter

java.io.FileWriter

See Also

Concepts

OutputStreamWriter Members

java.io Package

OutputStreamWriter Members

Provides an implementation of the abstract Writer class specific to writing characters to an OutputStream object.

The following tables list the members exposed by the OutputStreamWriter type.

Public Constructors

Name	Description
OutputStreamWriter	Overloaded. Initializes a new instance of an OutputStreamWriter object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer.(inherited from Writer)

Public Methods

Name	Description
close	Overridden. Closes the underlying OutputStream and releases all internal buffers.
equals	(inherited from Object)
flush	Overridden. Flushes the underlying OutputStream object.
getEncoding	Returns the character encoding being used on this instance.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Writes the next character or characters to the underlying OutputStream.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

OutputStreamWriter Class

Concepts

java.io Package

OutputStreamWriter Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer. (inherited from Writer)

See Also

Reference

OutputStreamWriter Class

Concepts

java.io Package

OutputStreamWriter Constructor

Initializes a new instance of an OutputStreamWriter object.

Overload List

Name	Description
OutputStreamWriter (OutputStream)	Initializes a new instance of an OutputStreamWriter object using the given OutputStream as the underlying stream.
OutputStreamWriter (OutputStream, String)	Initializes a new instance of an OutputStreamWriter object using the given OutputStream object as the underlying stream. Also specifies the character encoding of the underlying stream.

See Also

Reference

OutputStreamWriter Class

Concepts

OutputStreamWriter Members

java.io Package

OutputStreamWriter Constructor (OutputStream)

Initializes a new instance of an OutputStreamWriter object using the given OutputStream as the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.OutputStreamWriter(
    java.io.OutputStream out);
```

Parameters

out

The underlying output stream to write to.

See Also

Reference

OutputStreamWriter Class

Concepts

OutputStreamWriter Members

java.io Package

OutputStreamWriter Constructor (OutputStream, String)

Initializes a new instance of an OutputStreamWriter object using the given OutputStream object as the underlying stream. Also specifies the character encoding of the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.OutputStreamWriter(
    java.io.OutputStream out,
    java.lang.String enc) throws java.io.UnsupportedEncodingException;
```

Parameters

out

The underlying output stream to write to.

enc

The character encoding of the output stream (such as Unicode).

See Also

Reference

OutputStreamWriter Class

Concepts

OutputStreamWriter Members

java.io Package

OutputStreamWriter Methods

Public Methods

Name	Description
close	Overridden. Closes the underlying OutputStream and releases all internal buffers.
equals	(inherited from Object)
flush	Overridden. Flushes the underlying OutputStream object.
getEncoding	Returns the character encoding being used on this instance.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Writes the next character or characters to the underlying OutputStream.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

OutputStreamWriter Class

Concepts

java.io Package

OutputStreamWriter.close Method

Closes the underlying OutputStream and releases all internal buffers.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close an OutputStreamWriter object.

```
// outputstreamwriter_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a OutputStreamWriter object
            // attached to a ByteArrayOutputStream.
            ByteArrayOutputStream out =
                new ByteArrayOutputStream();
            OutputStreamWriter writer = new OutputStreamWriter(out);

            // Write to the output stream.
            String s = "Random bytes";
            writer.write(s);
            writer.flush();

            // Display the contents of the ByteArrayOutputStream.
            System.out.println(out.toString());

            // Close the OutputStreamWriter object.
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Random bytes
*/
```

See Also

Reference

OutputStreamWriter Class

Concepts

OutputStreamWriter Members

java.io Package

OutputStreamWriter.flush Method

Flushes the underlying OutputStream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush() throws java.io.IOException;
```

Example

The following example demonstrates how to flush the contents of an OutputStreamWriter buffer to the attached output stream.

```
// outputstreamwriter_flush.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a OutputStreamWriter object
            // attached to a ByteArrayOutputStream.
            ByteArrayOutputStream out =
                new ByteArrayOutputStream();
            OutputStreamWriter writer = new OutputStreamWriter(out);

            // Write to the output stream.
            String s = "Random bytes";
            writer.write(s);
            writer.flush();

            // Display the contents of the ByteArrayOutputStream.
            System.out.println(out.toString());

            // Close the OutputStreamWriter object.
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Random bytes
*/
```

See Also

Reference

OutputStreamWriter Class

Concepts

OutputStreamWriter Members

java.io Package

OutputStreamWriter.getEncoding Method

Returns the character encoding being used on this instance.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getEncoding();
```

Return Value

The character encoding of the underlying OutputStream object.

Example

The following example demonstrates how to determine the encoding that is being used by the output stream.

```
// outputstreamwriter_getencoding.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a OutputStreamWriter object
            // attached to a ByteArrayOutputStream.
            ByteArrayOutputStream out =
                new ByteArrayOutputStream();
            OutputStreamWriter writer = new OutputStreamWriter(out);

            // Write to the output stream.
            String s = "Random bytes";
            writer.write(s);
            writer.flush();

            // Display the contents of the ByteArrayOutputStream.
            System.out.println(out.toString());

            // Display the encoding being used.
            System.out.println("encoding: " +
                writer.getEncoding());

            // Close the OutputStreamWriter object.
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Random bytes
encoding: Cp1252
*/
```

See Also

Reference

OutputStreamWriter Class
Concepts
OutputStreamWriter Members
java.io Package

OutputStreamWriter.write Method

Writes the next character or characters to the underlying OutputStream.

Overload List

Name	Description
OutputStreamWriter.write (char[])	Places the given array of characters into the internal buffer holding the characters.
OutputStreamWriter.write (int)	Writes the next character to the underlying OutputStream.
OutputStreamWriter.write (String)	Places the given string into the internal buffer holding characters.
OutputStreamWriter.write (char[], int, int)	Writes count characters from the provided buffer, starting at an offset, to the underlying OutputStream object.
OutputStreamWriter.write (String, int, int)	Writes count characters from the provided string, starting at an offset, to the underlying OutputStream object.

See Also

Reference

OutputStreamWriter Class

Concepts

OutputStreamWriter Members

java.io Package

OutputStreamWriter.write Method (Int32)

Writes the next character to the underlying OutputStream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int oneChar) throws java.io.IOException;
```

Parameters

oneChar

The character to write to the underlying OutputStream object.

Example

The following example demonstrates how to write a single character at a time to the underlying output stream.

```
// outputstreamwriter_write.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a OutputStreamWriter object
            // attached to a ByteArrayOutputStream.
            ByteArrayOutputStream out =
                new ByteArrayOutputStream();
            OutputStreamWriter writer = new OutputStreamWriter(out);

            // Write to the output stream.
            String s = "Random bytes";
            for (int i = 0; i < s.length(); i++)
            {
                writer.write(s.charAt(i));
            }
            writer.flush();

            // Display the contents of the ByteArrayOutputStream.
            System.out.println(out.toString());

            // Close the OutputStreamWriter object.
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Random bytes
*/
```

See Also

Reference

OutputStreamWriter Class
Concepts
OutputStreamWriter Members
java.io Package

OutputStreamWriter.write Method (Char[ ], Int32, Int32)

Writes count characters from the provided buffer, starting at an offset, to the underlying OutputStream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    char[] buf,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buf

A buffer of characters to write to the underlying OutputStream.

offset

An offset into the provided buffer. This value represents the index of the first character in the buffer to be read. This value must be greater than zero.

count

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the provided buffer.

Example

The following example demonstrates how to write the contents of an array to the underlying output stream.

```
// outputstreamwriter_write_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a OutputStreamWriter object
            // attached to a ByteArrayOutputStream.
            ByteArrayOutputStream out =
                new ByteArrayOutputStream();
            OutputStreamWriter writer = new OutputStreamWriter(out);

            // Write to the output stream.
            String s = "Random bytes";
            char[] arr = s.toCharArray();
            // Only write from the 7th character on.
            writer.write(arr, 7, arr.length - 7);
            writer.flush();

            // Display the contents of the ByteArrayOutputStream.
            System.out.println(out.toString());

            // Close the OutputStreamWriter object.
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }
}

/*
Output:
bytes
*/
```

See Also

Reference

OutputStreamWriter Class

Concepts

OutputStreamWriter Members

java.io Package

OutputStreamWriter.write Method (String, Int32, Int32)

Writes count characters from the provided string, starting at an offset, to the underlying OutputStream object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

str

A string to write to the underlying OutputStream.

offset

An offset into the provided string. This value represents the index of the first character in the string to be read. This value must be greater than zero.

count

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the provided string.

Example

The following example demonstrates how to write the contents of a string to the underlying output stream.

```
// outputstreamwriter_write_3.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a OutputStreamWriter object
            // attached to a ByteArrayOutputStream.
            ByteArrayOutputStream out =
                new ByteArrayOutputStream();
            OutputStreamWriter writer = new OutputStreamWriter(out);

            // Write to the output stream.
            String s = "Random bytes";
            // Only write from the 7th character on.
            writer.write(s, 7, s.length() - 7);
            writer.flush();

            // Display the contents of the ByteArrayOutputStream.
            System.out.println(out.toString());

            // Close the OutputStreamWriter object.
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
```

```
}

/*
Output:
bytes
*/
```

See Also

Reference

OutputStreamWriter Class

Concepts

OutputStreamWriter Members

java.io Package

PipedInputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.PipedInputStream
    extends java.io.InputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

   java.io.PipedInputStream

See Also

Concepts

PipedInputStream Members

java.io Package

PipedInputStream Members

The following tables list the members exposed by the PipedInputStream type.

Public Constructors

Name	Description
PipedInputStream	Overloaded.

Public Fields

Name	Description
buffer	
in	
out	
PIPE_SIZE	

Public Methods

Name	Description
available	Overridden.
close	Overridden.
connect	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
receive	
reset	(inherited from InputStream)
skip	(inherited from InputStream)
toString	(inherited from InputStream)

Protected Methods

Name	Description

finalize	(inherited from Object)

See Also

Reference

PipedInputStream Class

Concepts

java.io Package

PipedInputStream Fields

Public Fields

Name	Description
buffer	
in	
out	
PIPE_SIZE	

See Also

Reference

PipedInputStream Class

Concepts

java.io Package

PipedInputStream.buffer Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected byte[] buffer;
```

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream.in Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int in;
```

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream.out Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int out;
```

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream.PIPE_SIZE Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected static final int PIPE_SIZE;
```

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream Constructor

Overload List

Name	Description
PipedInputStream ()	
PipedInputStream (PipedOutputStream)	

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream Constructor ()

Initializes a new instance of the PipedInputStream Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PipedInputStream();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream Constructor (PipedOutputStream)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PipedInputStream(
    java.io.PipedOutputStream src) throws java.io.IOException;
```

Parameters

src

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream Methods

Public Methods

Name	Description
available	Overridden.
close	Overridden.
connect	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
receive	
reset	(inherited from InputStream)
skip	(inherited from InputStream)
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PipedInputStream Class

Concepts

java.io Package

PipedInputStream.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int available() throws java.io.IOException;
```

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream.connect Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void connect(
    java.io.PipedOutputStream src) throws java.io.IOException;
```

Parameters

src

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream.read Method

Overload List

Name	Description
PipedInputStream.read ()	
PipedInputStream.read (byte[])	
PipedInputStream.read (byte[], int, int)	

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int read() throws java.io.IOException;
```

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream.read Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int read(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedInputStream.receive Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected synchronized void receive(
    int b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

PipedInputStream Class

Concepts

PipedInputStream Members

java.io Package

PipedOutputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.PipedOutputStream
    extends java.io.OutputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.OutputStream

   java.io.PipedOutputStream

See Also

Concepts

PipedOutputStream Members

java.io Package

PipedOutputStream Members

The following tables list the members exposed by the PipedOutputStream type.

Public Constructors

Name	Description
PipedOutputStream	Overloaded.

Public Methods

Name	Description
close	Overridden.
connect	
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PipedOutputStream Class

Concepts

java.io Package

PipedOutputStream Constructor

Overload List

Name	Description
PipedOutputStream ()	
PipedOutputStream (PipedInputStream)	

See Also

Reference

PipedOutputStream Class

Concepts

PipedOutputStream Members

java.io Package

PipedOutputStream Constructor ()

Initializes a new instance of the PipedOutputStream Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PipedOutputStream();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

PipedOutputStream Class

Concepts

PipedOutputStream Members

java.io Package

PipedOutputStream Constructor (PipedInputStream)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PipedOutputStream(
    java.io.PipedInputStream dest) throws java.io.IOException;
```

Parameters

dest

See Also
Reference
PipedOutputStream Class
Concepts
PipedOutputStream Members
java.io Package

PipedOutputStream Methods

Public Methods

Name	Description
close	Overridden.
connect	
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PipedOutputStream Class

Concepts

java.io Package

PipedOutputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

PipedOutputStream Class

Concepts

PipedOutputStream Members

java.io Package

PipedOutputStream.connect Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void connect(
    java.io.PipedInputStream dest) throws java.io.IOException;
```

Parameters

dest

See Also
Reference
PipedOutputStream Class
Concepts
PipedOutputStream Members
java.io Package

PipedOutputStream.flush Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void flush() throws java.io.IOException;
```

See Also

Reference

PipedOutputStream Class

Concepts

PipedOutputStream Members

java.io Package

PipedOutputStream.write Method

Overload List

Name	Description
PipedOutputStream.write (int)	
PipedOutputStream.write (byte[])	
PipedOutputStream.write (byte[], int, int)	

See Also

Reference

PipedOutputStream Class

Concepts

PipedOutputStream Members

java.io Package

PipedOutputStream.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int b) throws java.io.IOException;
```

Parameters
b

See Also
Reference
PipedOutputStream Class
Concepts
PipedOutputStream Members
java.io Package

PipedOutputStream.write Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

PipedOutputStream Class

Concepts

PipedOutputStream Members

java.io Package

PipedReader Class

Provides an implementation of the abstract Reader class specific to reading characters from a PipedWriter object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.PipedReader
    extends java.io.Reader
```

Example

The following example demonstrates the close, connect, read, and ready methods of the PipedReader class.

```
// pipedreader_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedReader object.
            PipedReader reader = new PipedReader();

            // Connect the PipedReader to a PipedWriter object.
            PipedWriter writer = new PipedWriter();
            reader.connect(writer);

            // Read from the PipedWriter.
            writer.write("Hello, PipedReader!");

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

Inheritance Hierarchy

java.lang.Object

  java.io.Reader

   java.io.PipedReader

See Also
Concepts
PipedReader Members
java.io Package

PipedReader Members

Provides an implementation of the abstract Reader class specific to reading characters from a PipedWriter object.

The following tables list the members exposed by the PipedReader type.

Public Constructors

Name	Description
PipedReader	Overloaded. Initializes a new instance of a PipedReader object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader.(inherited from Reader)

Public Methods

Name	Description
close	Overridden. Closes the connection between the PipedReader and PipedWriter.
connect	Opens the connection between a PipedReader and the PipedWriter.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Sets the mark to the next character of the characters to be read. (inherited from Reader)
markSupported	Determines whether the current position of the character can be marked. (inherited from Reader)
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the attached PipedWriter.
ready	Overridden. Determines whether the attached PipedWriter is ready for reading.
reset	Resets the reader for the characters such that the next character read is the character that is marked. (inherited from Reader)
skip	Skips over the next count characters of the characters being read or until the end of the array, whichever is less. (inherited from Reader)
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PipedReader Class
Concepts
java.io Package

PipedReader Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader. (inherited from Reader)

See Also

Reference

PipedReader Class

Concepts

java.io Package

PipedReader Constructor

Initializes a new instance of a PipedReader object.

Overload List

Name	Description
PipedReader ()	Initializes a new instance of a PipedReader object.
PipedReader (PipedWriter)	Initializes a new instance of a PipedReader object using the provided PipedWriter object as the source to be read.

See Also

Reference

PipedReader Class

Concepts

PipedReader Members

java.io Package

PipedReader Constructor ()

Initializes a new instance of a PipedReader object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PipedReader();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

PipedReader Class

Concepts

PipedReader Members

java.io Package

PipedReader Constructor (PipedWriter)

Initializes a new instance of a PipedReader object using the provided PipedWriter object as the source to be read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PipedReader(
    java.io.PipedWriter src) throws java.io.IOException;
```

Parameters

src

A PipedWriter object representing the source of the data to be read.

See Also

Reference

PipedReader Class

Concepts

PipedReader Members

java.io Package

PipedReader Methods

Public Methods

Name	Description
close	Overridden. Closes the connection between the PipedReader and PipedWriter.
connect	Opens the connection between a PipedReader and the PipedWriter.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Sets the mark to the next character of the characters to be read. (inherited from Reader)
markSupported	Determines whether the current position of the character can be marked. (inherited from Reader)
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the attached PipedWriter.
ready	Overridden. Determines whether the attached PipedWriter is ready for reading.
reset	Resets the reader for the characters such that the next character read is the character that is marked. (inherited from Reader)
skip	Skips over the next count characters of the characters being read or until the end of the array, whichever is less. (inherited from Reader)
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PipedReader Class

Concepts

java.io Package

PipedReader.close Method

Closes the connection between the PipedReader and PipedWriter.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close a PipedReader object.

```
// pipedreader_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedReader object.
            PipedReader reader = new PipedReader();

            // Connect the PipedReader to a PipedWriter object.
            PipedWriter writer = new PipedWriter();
            reader.connect(writer);

            // Read from the PipedWriter.
            writer.write("Hello, PipedReader!");

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

See Also

Reference

PipedReader Class

Concepts

PipedReader Members

java.io Package

PipedReader.connect Method

Opens the connection between a PipedReader and the PipedWriter.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void connect(
    java.io.PipedWriter src) throws java.io.IOException;
```

Parameters

src

A PipedWriter object representing the source of the data to be read.

Example

The following example shows how to connect a PipedReader object to a PipedWriter object.

```
// pipedreader_connect.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedReader object.
            PipedReader reader = new PipedReader();

            // Connect the PipedReader to a PipedWriter object.
            PipedWriter writer = new PipedWriter();
            reader.connect(writer);

            // Read from the PipedWriter.
            writer.write("Hello, PipedReader!");

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

Remarks

Opens a connection to the specified PipedWriter if this reader and the specified writer are not already connected. This method

fails by throwing an IOException exception if either the reader or writer are closed or there is an existing connection.

See Also

Reference

PipedReader Class

Concepts

PipedReader Members

java.io Package

PipedReader.read Method

Reads the next character or characters from the attached PipedWriter.

Overload List

Name	Description
PipedReader.read ()	Reads the next character from the attached PipedWriter.
PipedReader.read (char[])	Reads the characters in the array starting at an offset for the length specified and stores them in the provided array of characters.
PipedReader.read (char[], int, int)	Reads len characters from the attached PipedWriter and stores the characters in the provided buffer, starting from an offset.

See Also

Reference

PipedReader Class

Concepts

PipedReader Members

java.io Package

PipedReader.read Method ()

Reads the next character from the attached PipedWriter.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

Return Value

The numeric value of the character read from the writer, or -1 if either the writer or the connection is closed.

Example

The following example shows how to read one character at a time from a PipedWriter object.

```
// pipedreader_read.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedReader object.
            PipedReader reader = new PipedReader();

            // Connect the PipedReader to a PipedWriter object.
            PipedWriter writer = new PipedWriter();
            reader.connect(writer);

            // Read from the PipedWriter.
            writer.write("Hello, PipedReader!");

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

See Also

Reference

PipedReader Class

Concepts

PipedReader Members

PipedReader.read Method (Char[ ], Int32, Int32)

Reads len characters from the attached PipedWriter and stores the characters in the provided buffer, starting from an offset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

A buffer to store the characters read from the attached PipedWriter.

off

An offset into the array of characters. This value represents the index of the first character in the array to be written. This value must be greater than zero.

len

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the buffer.

Return Value

The number of characters read from the attached PipedWriter.

Example

The following example shows how to read data from a PipedWriter object and store the contents in an array.

```
// pipedreader_read_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedReader object.
            PipedReader reader = new PipedReader();

            // Connect the PipedReader to a PipedWriter object.
            PipedWriter writer = new PipedWriter();
            reader.connect(writer);

            // Read from the PipedWriter.
            String s = "Hello, PipedReader!";
            writer.write(s);

            char[] arr = new char[s.length()];
            while (reader.ready())
            {
                reader.read(arr, 0, arr.length);
                System.out.print(arr);
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
```

```
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

See Also

Reference

PipedReader Class

Concepts

PipedReader Members

java.io Package

PipedReader.ready Method

Determines whether the attached PipedWriter is ready for reading.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean ready() throws java.io.IOException;
```

Return Value

true if the attached PipedWriter object is ready for reading; false otherwise.

Example

The following example demonstrates how to determine if a PipedReader object is ready for reading.

```
// pipedreader_ready.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedReader object.
            PipedReader reader = new PipedReader();

            // Connect the PipedReader to a PipedWriter object.
            PipedWriter writer = new PipedWriter();
            reader.connect(writer);

            // Read from the PipedWriter.
            String s = "Hello, PipedReader!";
            writer.write(s);

            char[] arr = new char[s.length()];
            while (reader.ready())
            {
                reader.read(arr, 0, arr.length);
                System.out.print(arr);
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

See Also

Reference

PipedReader Class
Concepts
PipedReader Members
java.io Package

PipedWriter Class

Provides an implementation of the abstract Writer class specific to writing characters to a PipedWriter object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.PipedWriter
    extends java.io.Writer
```

Example

The following example demonstrates the close, connect, flush, and write methods of the PipedWriter class.

```
// pipedwriter_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedWriter object.
            PipedWriter writer = new PipedWriter();

            // Connect the PipedWriter to a PipedReader object.
            PipedReader reader = new PipedReader();
            writer.connect(reader);

            // Read from the PipedWriter.
            writer.write("Hello, PipedReader!");
            writer.flush();

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

Inheritance Hierarchy

java.lang.Object

  java.io.Writer

   java.io.PipedWriter

See Also
Concepts
PipedWriter Members
java.io Package

PipedWriter Members

Provides an implementation of the abstract Writer class specific to writing characters to a PipedWriter object.

The following tables list the members exposed by the PipedWriter type.

Public Constructors

Name	Description
PipedWriter	Overloaded. Initializes a new instance of a PipedWriter object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer.(inherited from Writer)

Public Methods

Name	Description
close	Overridden. Closes the connection between the PipedWriter and PipedReader.
connect	Opens the connection between a PipedWriter and the PipedReader.
equals	(inherited from Object)
flush	Overridden. Flushes the contents of the underlying buffer to the PipedReader.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Writes the next character or characters to the attached PipedReader.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PipedWriter Class

Concepts

java.io Package

PipedWriter Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer. (inherited from Writer)

See Also

Reference

PipedWriter Class

Concepts

java.io Package

PipedWriter Constructor

Initializes a new instance of a PipedWriter object.

Overload List

Name	Description
PipedWriter ()	Initializes a new instance of a PipedWriter object.
PipedWriter (PipedReader)	Initializes a new instance of a PipedWriter object using the provided PipedReader object as the destination for the characters to be written.

See Also

Reference

PipedWriter Class

Concepts

PipedWriter Members

java.io Package

PipedWriter Constructor ()

Initializes a new instance of a PipedWriter object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PipedWriter();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

PipedWriter Class

Concepts

PipedWriter Members

java.io Package

PipedWriter Constructor (PipedReader)

Initializes a new instance of a PipedWriter object using the provided PipedReader object as the destination for the characters to be written.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PipedWriter(
    java.io.PipedReader dest) throws java.io.IOException;
```

Parameters

dest

A PipedReader object representing the destination of the data to be written.

See Also

Reference

PipedWriter Class

Concepts

PipedWriter Members

java.io Package

PipedWriter Methods

Public Methods

Name	Description
close	Overridden. Closes the connection between the PipedWriter and PipedReader.
connect	Opens the connection between a PipedWriter and the PipedReader.
equals	(inherited from Object)
flush	Overridden. Flushes the contents of the underlying buffer to the PipedReader.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Writes the next character or characters to the attached PipedReader.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PipedWriter Class

Concepts

java.io Package

PipedWriter.close Method

Closes the connection between the PipedWriter and PipedReader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close a PipedWriter object.

```
// pipedwriter_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedWriter object.
            PipedWriter writer = new PipedWriter();

            // Connect the PipedWriter to a PipedReader object.
            PipedReader reader = new PipedReader();
            writer.connect(reader);

            // Read from the PipedWriter.
            writer.write("Hello, PipedReader!");
            writer.flush();

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

See Also

Reference

PipedWriter Class

Concepts

PipedWriter Members

java.io Package

PipedWriter.connect Method

Opens the connection between a PipedWriter and the PipedReader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void connect(
    java.io.PipedReader dest) throws java.io.IOException;
```

Parameters

dest

A PipedReader object representing the destination of the data to be written.

Example

The following example demonstrates how to connect a PipedWriter object to a PipedReader object.

```
// pipedwriter_connect.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedWriter object.
            PipedWriter writer = new PipedWriter();

            // Connect the PipedWriter to a PipedReader object.
            PipedReader reader = new PipedReader();
            writer.connect(reader);

            // Read from the PipedWriter.
            writer.write("Hello, PipedReader!");
            writer.flush();

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

See Also

Reference

PipedWriter Class
Concepts
PipedWriter Members
java.io Package

PipedWriter.flush Method

Flushes the contents of the underlying buffer to the PipedReader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush() throws java.io.IOException;
```

Example

The following example demonstrates how to flush the PipedWriter buffer.

```
// pipedwriter_flush.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedWriter object.
            PipedWriter writer = new PipedWriter();

            // Connect the PipedWriter to a PipedReader object.
            PipedReader reader = new PipedReader();
            writer.connect(reader);

            // Read from the PipedWriter.
            writer.write("Hello, PipedReader!");
            writer.flush();

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

See Also

Reference

PipedWriter Class

Concepts

PipedWriter Members

java.io Package

PipedWriter.write Method

Writes the next character or characters to the attached PipedReader.

Overload List

Name	Description
PipedWriter.write (char[])	Places the given array of characters into the internal buffer holding the characters.
PipedWriter.write (int)	Writes the next character to the attached PipedReader.
PipedWriter.write (String)	Places the given string into the internal buffer holding characters.
PipedWriter.write (char[], int, int)	Writes len characters from the provided buffer, starting at an offset, to the attached PipedReader.
PipedWriter.write (String, int, int)	Places the given string, starting from the offset and for the length specified, into the internal buffer holding the characters.

See Also

Reference

PipedWriter Class

Concepts

PipedWriter Members

java.io Package

PipedWriter.write Method (Int32)

Writes the next character to the attached PipedReader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int b) throws java.io.IOException;
```

Parameters

b

The character to write to the attached PipedReader.

Example

The following example shows how to write one character at a time to a PipedWriter object.

```
// pipedwriter_write.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedWriter object.
            PipedWriter writer = new PipedWriter();

            // Connect the PipedWriter to a PipedReader object.
            PipedReader reader = new PipedReader();
            writer.connect(reader);

            // Read from the PipedWriter.
            String s = "Hello, PipedReader!";
            for (int i = 0; i < s.length(); i++)
            {
                writer.write(s.charAt(i));
            }
            writer.flush();

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Hello, PipedReader!
*/
```

See Also

Reference

PipedWriter Class

Concepts

PipedWriter Members

java.io Package

PipedWriter.write Method (Char[ ], Int32, Int32)

Writes len characters from the provided buffer, starting at an offset, to the attached PipedReader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

A buffer of characters to write to the attached PipedReader.

off

An offset into the provided buffer. This value represents the index of the first character in the buffer to be read. This value must be greater than zero.

len

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the provided buffer.

Example

The following example demonstrates how to write an array of data to a PipedWriter object.

```
// pipedwriter_write_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PipedWriter object.
            PipedWriter writer = new PipedWriter();

            // Connect the PipedWriter to a PipedReader object.
            PipedReader reader = new PipedReader();
            writer.connect(reader);

            // Read from the PipedWriter.
            String s = "Hello, PipedReader!";
            char[] arr = s.toCharArray();
            writer.write(arr, 0, arr.length);
            writer.flush();

            while (reader.ready())
            {
                System.out.print((char)reader.read());
            }

            // Close the PipedReader and PipedWriter.
            reader.close();
            writer.close();
        }
        catch (IOException ex)
```

```
            {
                System.out.println(ex.toString());
            }
        }
    }

    /*
    Output:
    Hello, PipedReader!
    */
```

See Also

Reference

PipedWriter Class

Concepts

PipedWriter Members

java.io Package

PrintStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.PrintStream
    extends java.io.FilterOutputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.OutputStream

    java.io.FilterOutputStream

      java.io.PrintStream

See Also

Concepts

PrintStream Members

java.io Package

PrintStream Members

The following tables list the members exposed by the PrintStream type.

Public Constructors

Name	Description
PrintStream	Overloaded.

Public Fields

Name	Description
out	(inherited from FilterOutputStream)

Public Methods

Name	Description
checkError	
close	Overridden.
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
print	Overloaded.
println	Overloaded.
setError	
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PrintStream Class

Concepts

java.io Package

PrintStream Fields

Public Fields

Name	Description
out	(inherited from FilterOutputStream)

See Also

Reference

PrintStream Class

Concepts

java.io Package

PrintStream Constructor

Overload List

Name	Description
PrintStream (OutputStream)	
PrintStream (OutputStream, boolean)	

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream Constructor (OutputStream)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PrintStream(
    java.io.OutputStream out);
```

Parameters

out

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream Constructor (OutputStream, Boolean)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PrintStream(
    java.io.OutputStream out,
    boolean autoflush);
```

Parameters

out

autoflush

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream Methods

Public Methods

Name	Description
checkError	
close	Overridden.
equals	(inherited from Object)
flush	Overridden.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	(inherited from OutputStream)
print	Overloaded.
println	Overloaded.
setError	
toString	(inherited from OutputStream)
write	Overloaded. Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PrintStream Class

Concepts

java.io Package

PrintStream.checkError Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean checkError();
```

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close();
```

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.flush Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush();
```

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method

Overload List

Name	Description
PrintStream.print (boolean)	
PrintStream.print (char)	
PrintStream.print (char[])	
PrintStream.print (double)	
PrintStream.print (int)	
PrintStream.print (long)	
PrintStream.print (Object)	
PrintStream.print (float)	
PrintStream.print (String)	

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method (Boolean)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    boolean b);
```

Parameters

b

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method (Char)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    char c);
```

Parameters

c

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method (Char[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    char[] s);
```

Parameters

s

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method (Double)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    double d);
```

Parameters

d

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    int i);
```

Parameters

i

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method (Int64)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    long l);
```

Parameters

l

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method (Object)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    java.lang.Object o);
```

Parameters

o

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method (Single)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    float f);
```

Parameters

f

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.print Method (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    java.lang.String str);
```

Parameters

str

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.println Method

Overload List

Name	Description
PrintStream.println ()	
PrintStream.println (boolean)	
PrintStream.println (char)	
PrintStream.println (char[])	
PrintStream.println (double)	
PrintStream.println (int)	
PrintStream.println (long)	
PrintStream.println (Object)	
PrintStream.println (float)	
PrintStream.println (String)	

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.println Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println();
```

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.println Method (Boolean)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    boolean b);
```

Parameters

b

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.println Method (Char)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    char c);
```

Parameters

c

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.println Method (Char[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    char[] arry);
```

Parameters

arry

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.println Method (Double)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    double d);
```

Parameters

d

See Also
Reference
PrintStream Class
Concepts
PrintStream Members
java.io Package

PrintStream.println Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    int i);
```

Parameters
i

See Also
Reference
PrintStream Class
Concepts
PrintStream Members
java.io Package

PrintStream.println Method (Int64)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    long l);
```

Parameters

l

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.println Method (Object)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    java.lang.Object o);
```

Parameters

o

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.println Method (Single)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    float f);
```

Parameters

f

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.println Method (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    java.lang.String s);
```

Parameters

s

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.setError Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void setError();
```

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.write Method

Overload List

Name	Description
PrintStream.write (int)	
PrintStream.write (byte[])	
PrintStream.write (byte[], int, int)	

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintStream.write Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int b);
```

Parameters

b

See Also
Reference
PrintStream Class
Concepts
PrintStream Members
java.io Package

PrintStream.write Method (SByte[], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] b,
    int off,
    int len);
```

Parameters

b

off

len

See Also

Reference

PrintStream Class

Concepts

PrintStream Members

java.io Package

PrintWriter Class

Provides an implementation of the abstract Writer class specific to printing various primitive types and other objects to an underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.PrintWriter
    extends java.io.Writer
```

Example

The following example demonstrates the checkError, close, flush, and println methods of the PrintWriter class.

```
// printwriter_overview.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        printWriter.println(rand.nextInt());
        printWriter.println(rand.nextLong());
        printWriter.println(rand.nextFloat());
        printWriter.println(rand.nextDouble());

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
919498402
6438634853843200856
0.220032334
0.07553678009202236
*/
```

Inheritance Hierarchy

java.lang.Object

  java.io.Writer

    java.io.PrintWriter

See Also

Concepts

PrintWriter Members

java.io Package

PrintWriter Members

Provides an implementation of the abstract Writer class specific to printing various primitive types and other objects to an underlying output stream.

The following tables list the members exposed by the PrintWriter type.

Public Constructors

Name	Description
PrintWriter	Overloaded. Initializes a new instance of a PrintWriter object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer.(inherited from Writer)

Public Methods

Name	Description
checkError	Determines whether there were any errors or exceptions when printing to the underlying output stream.
close	Overridden. Closes the underlying output stream and releases all internal buffers.
equals	(inherited from Object)
flush	Overridden. Flushes the underlying output stream.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
print	Overloaded. Prints various types to the underlying output stream.
println	Overloaded. Prints various types to the underlying output stream, followed by a newline character.
setError	Sets a value indicating that an error or exception has occurred.
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Overridden. Writes the next character or characters to the underlying output stream.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PrintWriter Class

Concepts

java.io Package

PrintWriter Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer. (inherited from Writer)

See Also

Reference

PrintWriter Class

Concepts

java.io Package

PrintWriter Constructor

Initializes a new instance of a PrintWriter object.

Overload List

Name	Description
PrintWriter (OutputStream)	Initializes a new instance of a PrintWriter object using the given OutputStream as the underlying stream to print to.
PrintWriter (Writer)	Initializes a new instance of a PrintWriter object using the given Writer as the underlying stream to print to.
PrintWriter (OutputStream, boolean)	Initializes a new instance of a PrintWriter object using the given OutputStream as the underlying stream to print to. You can also specify whether the stream should be automatically flushed when printed to.
PrintWriter (Writer, boolean)	Initializes a new instance of a PrintWriter object using the given Writer as the underlying stream to print to. You can also specify whether the stream should be automatically flushed when printed to.

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter Constructor (OutputStream)

Initializes a new instance of a PrintWriter object using the given OutputStream as the underlying stream to print to.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PrintWriter(
    java.io.OutputStream out);
```

Parameters

out

The underlying output stream to print to.

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter Constructor (Writer)

Initializes a new instance of a PrintWriter object using the given Writer as the underlying stream to print to.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PrintWriter(
    java.io.Writer out);
```

Parameters

out

The underlying output stream to print to.

See Also
Reference
PrintWriter Class
Concepts
PrintWriter Members
java.io Package

PrintWriter Constructor (OutputStream, Boolean)

Initializes a new instance of a PrintWriter object using the given OutputStream as the underlying stream to print to. You can also specify whether the stream should be automatically flushed when printed to.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PrintWriter(
    java.io.OutputStream out,
    boolean autoflush);
```

Parameters

out

The underlying output stream to print to.

autoflush

A value indicating whether the stream should be automatically flushed when it is written to.

See Also
Reference
PrintWriter Class
Concepts
PrintWriter Members
java.io Package

PrintWriter Constructor (Writer, Boolean)

Initializes a new instance of a PrintWriter object using the given Writer as the underlying stream to print to. You can also specify whether the stream should be automatically flushed when printed to.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PrintWriter(
    java.io.Writer out,
    boolean autoflush);
```

Parameters

out

The underlying output stream to print to.

autoflush

A value indicating whether the stream should be automatically flushed when it is written to.

See Also
Reference
PrintWriter Class
Concepts
PrintWriter Members
java.io Package

PrintWriter Methods

Public Methods

Name	Description
checkError	Determines whether there were any errors or exceptions when printing to the underlying output stream.
close	Overridden. Closes the underlying output stream and releases all internal buffers.
equals	(inherited from Object)
flush	Overridden. Flushes the underlying output stream.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
print	Overloaded. Prints various types to the underlying output stream.
println	Overloaded. Prints various types to the underlying output stream, followed by a newline character.
setError	Sets a value indicating that an error or exception has occurred.
toString	Displays a human readable representation of a Writer object. (inherited from Writer)
write	Overloaded. Overridden. Writes the next character or characters to the underlying output stream.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PrintWriter Class

Concepts

java.io Package

PrintWriter.checkError Method

Determines whether there were any errors or exceptions when printing to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean checkError();
```

Return Value

true if there were any errors or exceptions when printing to the underlying output stream; false otherwise.

Example

The following example demonstrates how to call checkError to verify that the PrintWriter object successfully printed to the underlying output stream.

```
// printwriter_checkerror.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        printWriter.println(rand.nextInt());
        printWriter.println(rand.nextLong());
        printWriter.println(rand.nextFloat());
        printWriter.println(rand.nextDouble());

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
1783513883
4369932518214174912
0.4472599
0.41234255408466969
*/
```

See Also

Reference

PrintWriter Class
Concepts
PrintWriter Members
java.io Package

PrintWriter.close Method

Closes the underlying output stream and releases all internal buffers.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close();
```

Example

The following example demonstrates how to close a PrintWriter object once you have written to the underlying output stream.

```
// printwriter_close.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        printWriter.println(rand.nextInt());
        printWriter.println(rand.nextLong());
        printWriter.println(rand.nextFloat());
        printWriter.println(rand.nextDouble());

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
1067437389
5375348907260604722
0.3197248
0.92409548626073634
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.flush Method

Flushes the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush();
```

Example

The following example demonstrates how to flush the contents of a PrintWriter object to the underlying output stream.

```
// printwriter_flush.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        printWriter.println(rand.nextInt());
        printWriter.println(rand.nextLong());
        printWriter.println(rand.nextFloat());
        printWriter.println(rand.nextDouble());

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
-1795952035
-7904475002891073741
0.8822727
0.74217617902306976
*/
```

See Also

Reference

PrintWriter Class

Concepts
PrintWriter Members
java.io Package

PrintWriter.print Method

Prints various types to the underlying output stream.

Overload List

Name	Description
PrintWriter.print (boolean)	Prints a boolean value to the underlying output stream.
PrintWriter.print (char)	Prints a character to the underlying output stream.
PrintWriter.print (char[])	Prints an array of characters to the underlying output stream.
PrintWriter.print (double)	Prints a double value to the underlying output stream.
PrintWriter.print (int)	Prints an integer value to the underlying output stream.
PrintWriter.print (long)	Prints a long value to the underlying output stream.
PrintWriter.print (Object)	Prints an object to the underlying output stream.
PrintWriter.print (float)	Prints a float value to the underlying output stream.
PrintWriter.print (String)	Prints a String to the underlying output stream.

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.print Method (Boolean)

Prints a boolean value to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    boolean b);
```

Parameters

b

The boolean value to be printed.

Example

The following example demonstrates how to print random boolean values to a StringWriter buffer.

```
// printwriter_print_boolean.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.print(rand.nextBoolean());
            printWriter.println();
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
false
true
true
false
```

```
false
true
true
true
false
false
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.print Method (Char)

Prints a character to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    char c);
```

Parameters

c

The character to be printed.

Example

The following example demonstrates how to print random char values to a StringWriter buffer.

```
// printwriter_print_char.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            printWriter.print((char)randInt);
            printWriter.println();
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}
```

```
/*
Output:
;
0
^
f
J
0
=
:
P
a
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.print Method (Char[ ])

Prints an array of characters to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    char[] charArray);
```

Parameters

charArray

The array of characters to be printed.

Example

The following example demonstrates how to print a random array of char values to a StringWriter buffer.

```
// printwriter_print_chararray.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        char[] arr = new char[10];
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            arr[i] = (char)randInt;
        }
        printWriter.print(arr);

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
```

```
}

/*
Output:
&gVHdN!x-*
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.print Method (Double)

Prints a double value to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    double d);
```

Parameters

d

The double value to be printed.

Example

The following example demonstrates how to print random double values to a StringWriter buffer.

```
// printwriter_print_double.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.print(rand.nextDouble());
            printWriter.println();
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
0.22111552345988439
0.71412270820949275
0.46749730473515727
0.40253875268354533
```

```
0.45325243734855214
0.12178138237062242
0.33311483948047493
0.232486424424311
0.73214155747650689
0.2730524545489158
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.print Method (Int32)

Prints an integer value to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    int i);
```

Parameters

i

The integer value to be printed.

Example

The following example demonstrates how to print random int values to a StringWriter buffer.

```
// printwriter_print_int.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.print(rand.nextInt());
            printWriter.println();
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
1179473779
-1493859189
-1258719222
947110265
```

```
-1678818902
-138830361
905713311
-1925380613
466622335
621473841
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.print Method (Int64)

Prints a long value to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    long l);
```

Parameters

l

The long value to be printed.

Example

The following example demonstrates how to print random long values to a StringWriter buffer.

```
// printwriter_print_long.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.print(rand.nextLong());
            printWriter.println();
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
3786335069141929567
4160774848858038463
-1003712314565441972
6871459982539519038
```

```
8405481078492958439
-3542697386318197532
4465413492605614828
-2279875300762862514
-9041062491321340712
-2477215260312440059
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.print Method (Object)

Prints an object to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    java.lang.Object o);
```

Parameters

o

The object to be printed.

Example

The following example demonstrates how to print random Integer objects to a StringWriter buffer.

```
// printwriter_print_object.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        int randInt;
        for (int i = 0; i < 10; i++)
        {
            randInt = rand.nextInt();
            printWriter.print(new Integer(randInt));
            printWriter.println();
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
781902959
1843952855
```

```
-1116756940
285737205
473450705
117169121
-760428226
-1975219998
1253373131
930932395
*/
```

Remarks

The string printed is the same text that would be displayed if o.toString(); were called.

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.print Method (Single)

Prints a float value to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    float f);
```

Parameters

f

The float value to be printed.

Example

The following example demonstrates how to print random float values to a StringWriter buffer.

```
// printwriter_print_float.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.print(rand.nextFloat());
            printWriter.println();
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
0.444933
0.1618141
0.0215989351
0.118006468
```

```
0.75749886
0.262105
0.707075
0.9051153
0.463990152
0.588330448
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.print Method (String)

Prints a String to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void print(
    java.lang.String str);
```

Parameters

str

The string to be printed.

Example

The following example demonstrates how to print a string of random characters to a StringWriter buffer.

```
// printwriter_print_string.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        char[] arr = new char[10];
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            arr[i] = (char)randInt;
        }
        printWriter.print(new String(arr));

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
```

```
}

/*
Output:
oH0N5EC2Cb
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method

Prints various types to the underlying output stream, followed by a newline character.

Overload List

Name	Description
PrintWriter.println ()	Prints a newline character to the underlying output stream.
PrintWriter.println (boolean)	Prints a boolean value to the underlying output stream, followed by a newline character.
PrintWriter.println (char)	Prints a character to the underlying output stream, followed by a newline character.
PrintWriter.println (char[])	Prints an array of characters to the underlying output stream, followed by a newline character.
PrintWriter.println (double)	Prints a double value to the underlying output stream, followed by a newline character.
PrintWriter.println (int)	Prints an integer to the underlying output stream, followed by a newline character.
PrintWriter.println (long)	Prints a long value to the underlying output stream, followed by a newline character.
PrintWriter.println (Object)	Prints an object to the underlying output stream, followed by a newline character.
PrintWriter.println (float)	Prints a float value to the underlying output stream, followed by a newline character.
PrintWriter.println (String)	Prints a String to the underlying output stream, followed by a newline character.

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method ()

Prints a newline character to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println();
```

Example

The following example demonstrates how to print a newline character to a StringWriter buffer.

```
// printwriter_println.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.print(rand.nextInt());
            printWriter.println();
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
189228037
1655213393
1748306950
-1530658345
-865426539
-618026644
-1537214592
1504546217
310250331
```

```
-839698328
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method (Boolean)

Prints a boolean value to the underlying output stream, followed by a newline character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    boolean b);
```

Parameters

b

The boolean value to be printed.

Example

The following example demonstrates how to print random boolean values to a StringWriter buffer.

```
// printwriter_println_boolean.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.println(rand.nextBoolean());
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
true
true
true
false
false
```

```
false
true
false
true
true
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method (Char)

Prints a character to the underlying output stream, followed by a newline character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    char c);
```

Parameters

c

The character to be printed.

Example

The following example demonstrates how to print random char values to a StringWriter buffer.

```
// printwriter_println_char.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            printWriter.println((char)randInt);
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}
```

```
/*
Output:
'
s
$
(
D
3
i
=
F
o
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method (Char[ ])

Prints an array of characters to the underlying output stream, followed by a newline character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    char[] arry);
```

Parameters

arry

The array of characters to be printed.

Example

The following example demonstrates how to print an array of random char values to a StringWriter buffer.

```
// printwriter_println_chararray.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        char[] arr = new char[10];
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            arr[i] = (char)randInt;
        }
        printWriter.println(arr);

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
```

```
}

/*
Output:
oU#/i0V?f3
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method (Double)

Prints a double value to the underlying output stream, followed by a newline character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    double d);
```

Parameters

d

The double value to be printed.

Example

The following example demonstrates how to print random double values to a StringWriter buffer.

```
// printwriter_println_double.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.println(rand.nextDouble());
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
0.049098646526183121
0.57835846075847608
0.63727950028611535
0.25283226632606604
0.13250715455371021
```

```
0.76372010810678892
0.32138054288587736
0.27010703232034505
0.066136358456775679
0.46604961123715993
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method (Int32)

Prints an integer to the underlying output stream, followed by a newline character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    int i);
```

Parameters

i

The integer to be printed.

Example

The following example demonstrates how to print random int values to a StringWriter buffer.

```
// printwriter_println_int.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.println(rand.nextInt());
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
630611556
227991716
1481528284
-322415982
-80496314
```

```
1929473776
-1168890856
1621578169
-1915263323
-1987968255
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method (Int64)

Prints a long value to the underlying output stream, followed by a newline character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    long l);
```

Parameters

l

The long value to be printed.

Example

The following example demonstrates how to print random long values to a StringWriter buffer.

```
// printwriter_println_long.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.println(rand.nextLong());
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
2359469369572711718
-1752854823586231308
8313904213493496961
7062038769965299808
-1559504970770141700
```

```
1772345519805857303
919025676985800467
8064098131209990983
-8867870457877401800
8053236360516612856
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method (Object)

Prints an object to the underlying output stream, followed by a newline character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    java.lang.Object o);
```

Parameters

o

The object to be printed.

Example

The following example demonstrates how to print random Integer objects to a StringWriter buffer.

```
// printwriter_println_object.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        int randInt;
        for (int i = 0; i < 10; i++)
        {
            randInt = rand.nextInt();
            printWriter.println(new Integer(randInt));
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
-543431831
-950659014
-779293546
```

```
502965373
-341640195
-1880187005
1402244877
-1245040133
-874112932
734881471
*/
```

Remarks

The string printed is the same text that would be displayed if o.toString(); were called.

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method (Single)

Prints a float value to the underlying output stream, followed by a newline character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    float f);
```

Parameters

f

The float value to be printed.

Example

The following example demonstrates how to print random float values to a StringWriter buffer.

```
// printwriter_println_float.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        for (int i = 0; i < 10; i++)
        {
            printWriter.println(rand.nextFloat());
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
0.3931421
0.21162647
0.6578447
0.6726297
0.8801815
```

```
0.1248005
0.0174156427
0.554498732
0.1375094
0.5722707
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.println Method (String)

Prints a String to the underlying output stream, followed by a newline character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void println(
    java.lang.String s);
```

Parameters

s

The string to be printed.

Example

The following example demonstrates how to print a string of random characters to a StringWriter buffer.

```
// printwriter_println_string.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        char[] arr = new char[10];
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            arr[i] = (char)randInt;
        }
        printWriter.println(new String(arr));

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
```

```
}

/*
Output:
H.u91t5e&,
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.setError Method

Sets a value indicating that an error or exception has occurred.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected void setError();
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.write Method

Writes the next character or characters to the underlying output stream.

Overload List

Name	Description
PrintWriter.write (char[])	Writes the characters in the provided buffer to the underlying output stream.
PrintWriter.write (int)	Writes the next character to the underlying output stream.
PrintWriter.write (String)	Writes the provided String to the underlying output stream.
PrintWriter.write (char[], int, int)	Writes len characters from the provided buffer, starting at an offset, to the underlying output stream.
PrintWriter.write (String, int, int)	Writes len characters from the provided string, starting at an offset, to the underlying output s tream.

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.write Method (Char[ ])

Writes the characters in the provided buffer to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    char[] b);
```

Parameters

b

A buffer of characters to write to the underlying output stream.

Example

The following example demonstrates how to write an array of characters to the underlying output stream.

```
// printwriter_write.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        char[] arr = new char[10];
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            arr[i] = (char)randInt;
        }
        printWriter.write(arr);

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
```

```
}

/*
Output:
W:oy6,%]Fg
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.write Method (Int32)

Writes the next character to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int b);
```

Parameters

b

The character to write to the underlying output stream.

Example

The following example demonstrates how to write a single character to the underlying output stream.

```
// printwriter_write_2.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            printWriter.write((char)randInt);
            printWriter.println();
        }

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}
```

```
/*
Output:
"
"
R
6
"
d
y
0
P
;
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.write Method (String)

Writes the provided String to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str);
```

Parameters

str

The string to write to the underlying output stream.

Example

The following example demonstrates how to write a string to the underlying output stream.

```
// printwriter_write_3.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        char[] arr = new char[10];
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            arr[i] = (char)randInt;
        }
        printWriter.write(new String(arr));

        // Flush the buffer.
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
```

```
}

/*
Output:
]8$1XZsfpM
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.write Method (Char[ ], Int32, Int32)

Writes len characters from the provided buffer, starting at an offset, to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    char[] b,
    int off,
    int len);
```

Parameters

b

A buffer of characters to write to the underlying output stream.

off

An offset into the provided buffer. This value represents the index of the first character in the buffer to be read. This value must be greater than zero.

len

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the provided buffer.

Example

The following example demonstrates how to write an array of characters to the underlying output stream.

```
// printwriter_write_4.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        char[] arr = new char[10];
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            arr[i] = (char)randInt;
        }
        printWriter.write(arr, 0, arr.length);

        // Flush the buffer.
```

```
        printWriter.flush();

        // Display the contents of the StringWriter.
        if (!printWriter.checkError())
        {
            System.out.println(stringWriter.getBuffer());
        }

        // Close the PrintWriter and String Writer objects.
        stringWriter.close();
        printWriter.close();
    }
}

/*
Output:
an-dvmO,p
*/
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PrintWriter.write Method (String, Int32, Int32)

Writes len characters from the provided string, starting at an offset, to the underlying output stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str,
    int off,
    int len);
```

Parameters

str

A string to write to the underlying output stream.

off

An offset into the provided string. This value represents the index of the first character in the string to be read. This value must be greater than zero.

len

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the provided string.

Example

The following example demonstrates how to write a string to the underlying output stream.

```
// printwriter_write_5.jsl

import java.io.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new instance of a PrintWriter object along with
        // an underlying StringWriter object.
        StringWriter stringWriter = new StringWriter();
        PrintWriter printWriter = new PrintWriter(stringWriter);

        // Print some random data to the StringWriter.
        Date date = new Date();
        Random rand = new Random(date.getTime());

        // Characters are between 32 and 126 (ASCII).
        char[] arr = new char[10];
        for (int i = 0; i < 10; i++)
        {
            int randInt = 0;
            while (randInt < 32)
            {
                randInt = rand.nextInt(126);
            }

            arr[i] = (char)randInt;
        }
        printWriter.write(new String(arr), 0, arr.length);

        // Flush the buffer.
```

```
            printWriter.flush();

            // Display the contents of the StringWriter.
            if (!printWriter.checkError())
            {
                System.out.println(stringWriter.getBuffer());
            }

            // Close the PrintWriter and String Writer objects.
            stringWriter.close();
            printWriter.close();
        }
    }

    /*
    Output:
    2f6}Lc#B$F
    */
```

See Also

Reference

PrintWriter Class

Concepts

PrintWriter Members

java.io Package

PushbackInputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.PushbackInputStream
    extends java.io.FilterInputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

    java.io.FilterInputStream

      java.io.PushbackInputStream

See Also

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream Members

The following tables list the members exposed by the PushbackInputStream type.

Public Constructors

Name	Description
PushbackInputStream	Overloaded.

Public Fields

Name	Description
buf	
in	(inherited from FilterInputStream)
pos	

Public Methods

Name	Description
available	Overridden.
close	(inherited from FilterInputStream)
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from FilterInputStream)
markSupported	Overridden.
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	(inherited from FilterInputStream)
skip	(inherited from FilterInputStream)
toString	(inherited from InputStream)
unread	Overloaded.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PushbackInputStream Class
Concepts
java.io Package

PushbackInputStream Fields

Public Fields

Name	Description
buf	
in	(inherited from FilterInputStream)
pos	

See Also

Reference

PushbackInputStream Class

Concepts

java.io Package

PushbackInputStream.buf Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected byte[] buf;
```

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream.pos Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int pos;
```

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream Constructor

Overload List

Name	Description
PushbackInputStream (InputStream)	
PushbackInputStream (InputStream, int)	

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream Constructor (InputStream)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PushbackInputStream(
    java.io.InputStream in);
```

Parameters

in

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream Constructor (InputStream, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PushbackInputStream(
    java.io.InputStream in,
    int size);
```

Parameters

in

size

See Also
Reference
PushbackInputStream Class
Concepts
PushbackInputStream Members
java.io Package

PushbackInputStream Methods

Public Methods

Name	Description
available	Overridden.
close	(inherited from FilterInputStream)
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from FilterInputStream)
markSupported	Overridden.
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	(inherited from FilterInputStream)
skip	(inherited from FilterInputStream)
toString	(inherited from InputStream)
unread	Overloaded.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PushbackInputStream Class

Concepts

java.io Package

PushbackInputStream.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int available() throws java.io.IOException;
```

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream.markSupported Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream.read Method

Overload List

Name	Description
PushbackInputStream.read ()	
PushbackInputStream.read (byte[])	
PushbackInputStream.read (byte[], int, int)	

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream.read Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream.unread Method

Overload List

Name	Description
PushbackInputStream.unread (int)	
PushbackInputStream.unread (byte[])	
PushbackInputStream.unread (byte[], int, int)	

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream.unread Method (Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void unread(
    int b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream.unread Method (SByte[ ])

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void unread(
    byte[] b) throws java.io.IOException;
```

Parameters

b

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackInputStream.unread Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void unread(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

PushbackInputStream Class

Concepts

PushbackInputStream Members

java.io Package

PushbackReader Class

Provides an implementation of the abstract Reader class that buffers data from another stream. The internal buffer is initially filled with the contents read in from the underlying stream. Characters in this internal buffer can then be "pushed back", or replaced with other characters at the position marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.PushbackReader
    extends java.io.FilterReader
```

Example

The following example demonstrates the close, read, ready, and unread methods of the PushbackReader class.

```
// pushbackreader_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PushbackReader object with an
            // underlying StringReader buffer.
            String s = "This is the internal buffer of StringReader.";
            StringReader stringReader = new StringReader(s);
            PushbackReader pbReader = new PushbackReader(stringReader, 100);

            // Read from the underlying StringReader buffer.
            char[] arr = new char[s.length()];
            if (pbReader.ready())
            {
                pbReader.read(arr, 0, arr.length);
            }
            System.out.println(arr);

            // "Unread" some of the data read from the underlying
            // StringReader buffer.
            String newS = "Replacement internal buffer.";
            char[] newSArr = newS.toCharArray();
            pbReader.unread(newSArr);

            // Close the PushbackReader object and the underlying
            // StringReader object.
            stringReader.close();
            pbReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal buffer of StringReader.
*/
```

Inheritance Hierarchy

java.lang.Object

  java.io.Reader

    java.io.FilterReader

      java.io.PushbackReader

See Also

Concepts

PushbackReader Members

java.io Package

PushbackReader Members

Provides an implementation of the abstract Reader class that buffers data from another stream. The internal buffer is initially filled with the contents read in from the underlying stream. Characters in this internal buffer can then be "pushed back", or replaced with other characters at the position marked.

The following tables list the members exposed by the PushbackReader type.

Public Constructors

Name	Description
PushbackReader	Overloaded. Initializes a new instance of a PushbackReader object.

Public Fields

Name	Description
in	(inherited from FilterReader)
lock	The object used to synchronize operations on the reader.(inherited from Reader)

Public Methods

Name	Description
close	Overridden. Closes the stream being buffered and clears all internal buffers.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from FilterReader)
markSupported	Overridden. Determines whether the current position of the stream can be marked. Objects of type PushbackReader do not support marking.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the underlying stream.
ready	Overridden. Determines whether the underlying stream is ready for reading.
reset	(inherited from FilterReader)
skip	(inherited from FilterReader)
toString	Displays a human readable representation of the reader. (inherited from Reader)
unread	Overloaded. Replaces the characters that were read into the internal buffer from the underlying stream with the given character or characters.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PushbackReader Class

Concepts

java.io Package

PushbackReader Fields

Public Fields

Name	Description
in	(inherited from FilterReader)
lock	The object used to synchronize operations on the reader. (inherited from Reader)

See Also

Reference

PushbackReader Class

Concepts

java.io Package

PushbackReader Constructor

Initializes a new instance of a PushbackReader object.

Overload List

Name	Description
PushbackReader (Reader)	Initializes a new instance of a PushbackReader object. The provided Reader object is the source of the data to be buffered.
PushbackReader (Reader, int)	Initializes a new instance of a PushbackReader object. The provided Reader object is the source of the data to be buffered. The internal buffer is initializes to the given size.

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

PushbackReader Constructor (Reader)

Initializes a new instance of a PushbackReader object. The provided Reader object is the source of the data to be buffered.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PushbackReader(
    java.io.Reader in);
```

Parameters

in

A Reader object is the source of the data to be buffered.

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

PushbackReader Constructor (Reader, Int32)

Initializes a new instance of a PushbackReader object. The provided Reader object is the source of the data to be buffered. The internal buffer is initializes to the given size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.PushbackReader(
    java.io.Reader in,
    int size);
```

Parameters

in

A Reader object is the source of the data to be buffered.

size

The size of the internal buffer.

See Also
Reference
PushbackReader Class
Concepts
PushbackReader Members
java.io Package

PushbackReader Methods

Public Methods

Name	Description
close	Overridden. Closes the stream being buffered and clears all internal buffers.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from FilterReader)
markSupported	Overridden. Determines whether the current position of the stream can be marked. Objects of type PushbackReader do not support marking.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters from the underlying stream.
ready	Overridden. Determines whether the underlying stream is ready for reading.
reset	(inherited from FilterReader)
skip	(inherited from FilterReader)
toString	Displays a human readable representation of the reader. (inherited from Reader)
unread	Overloaded. Replaces the characters that were read into the internal buffer from the underlying stream with t he given character or characters.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PushbackReader Class

Concepts

java.io Package

PushbackReader.close Method

Closes the stream being buffered and clears all internal buffers.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close a PushbackReader object.

```
// pushbackreader_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PushbackReader object with an
            // underlying StringReader buffer.
            String s = "This is the internal buffer of StringReader.";
            StringReader stringReader = new StringReader(s);
            PushbackReader pbReader = new PushbackReader(stringReader, 100);

            // Read from the underlying StringReader buffer.
            char[] arr = new char[s.length()];
            pbReader.read(arr);
            System.out.println(arr);

            // "Unread" some of the data read from the underlying
            // StringReader buffer.
            String newS = "Replacement internal buffer.";
            char[] newSArr = newS.toCharArray();
            pbReader.unread(newSArr);

            // Close the PushbackReader object and the underlying
            // StringReader object.
            stringReader.close();
            pbReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal buffer of StringReader.
*/
```

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

PushbackReader.markSupported Method

Determines whether the current position of the stream can be marked. Objects of type PushbackReader do not support marking.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

Return Value

Always false.

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

PushbackReader.read Method

Reads the next character or characters from the underlying stream.

Overload List

Name	Description
PushbackReader.read ()	Reads the next character from the underlying stream.
PushbackReader.read (char[])	Reads the characters in the array starting at an offset for the length specified and stores them in the provided array of characters.
PushbackReader.read (char[], int, int)	Reads the characters from the underlying stream for the length specified, and stores them in the provided array of characters, starting at an offset.

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

PushbackReader.read Method ()

Reads the next character from the underlying stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

Return Value

The numeric value of the character read from the stream using the character encoding of the system (such as UTF-8).

Example

The following example demonstrates how to read a single character at a time from the underlying stream.

```
// pushbackreader_read.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PushbackReader object with an
            // underlying StringReader buffer.
            String s = "This is the internal buffer of StringReader.";
            StringReader stringReader = new StringReader(s);
            PushbackReader pbReader = new PushbackReader(stringReader, 100);

            // Read from the underlying StringReader buffer.
            for (int i = 0; i < s.length(); i++)
            {
                System.out.print((char)pbReader.read());
            }

            // "Unread" some of the data read from the underlying
            // StringReader buffer.
            String newS = "Replacement internal buffer.";
            char[] newSArr = newS.toCharArray();
            pbReader.unread(newSArr);

            // Close the PushbackReader object and the underlying
            // StringReader object.
            stringReader.close();
            pbReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal buffer of StringReader.
*/
```

See Also

Reference
PushbackReader Class
Concepts
PushbackReader Members
java.io Package

PushbackReader.read Method (Char[], Int32, Int32)

Reads the characters from the underlying stream for the length specified, and stores them in the provided array of characters, starting at an offset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

The array of characters to store the characters read from the underlying stream.

off

An offset into the array of characters. This value represents the index of the first character in the array to be written. This value must be greater than zero.

len

The number of characters to be read. This value plus the offset must be less than the overall length of the array.

Return Value

The number of characters read.

Example

The following example demonstrates how to read data from the underlying stream and store the data in an array.

```
// pushbackreader_read_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PushbackReader object with an
            // underlying StringReader buffer.
            String s = "This is the internal buffer of StringReader.";
            StringReader stringReader = new StringReader(s);
            PushbackReader pbReader = new PushbackReader(stringReader, 100);

            // Read from the underlying StringReader buffer.
            char[] arr = new char[s.length()];
            pbReader.read(arr, 0, arr.length);
            System.out.println(arr);

            // "Unread" some of the data read from the underlying
            // StringReader buffer.
            String newS = "Replacement internal buffer.";
            char[] newSArr = newS.toCharArray();
            pbReader.unread(newSArr);

            // Close the PushbackReader object and the underlying
            // StringReader object.
            stringReader.close();
```

```
            pbReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal buffer of StringReader.
*/
```

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

PushbackReader.ready Method

Determines whether the underlying stream is ready for reading.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean ready() throws java.io.IOException;
```

Return Value

true if the underlying stream is ready for reading; false otherwise.

Example

The following example demonstrates how to determine if a PushbackReader object is ready for reading.

```
// pushbackreader_ready.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PushbackReader object with an
            // underlying StringReader buffer.
            String s = "This is the internal buffer of StringReader.";
            StringReader stringReader = new StringReader(s);
            PushbackReader pbReader = new PushbackReader(stringReader, 100);

            // Read from the underlying StringReader buffer.
            char[] arr = new char[s.length()];
            if (pbReader.ready())
            {
                pbReader.read(arr, 0, arr.length);
            }
            System.out.println(arr);

            // "Unread" some of the data read from the underlying
            // StringReader buffer.
            String newS = "Replacement internal buffer.";
            char[] newSArr = newS.toCharArray();
            pbReader.unread(newSArr);

            // Close the PushbackReader object and the underlying
            // StringReader object.
            stringReader.close();
            pbReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal buffer of StringReader.
*/
```

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

PushbackReader.unread Method

Replaces the characters that were read into the internal buffer from the underlying stream with the given character or characters.

Overload List

Name	Description
PushbackReader.unread (char[])	Replaces the characters read into the internal buffer from the underlying stream with the given array of characters. The characters are replaced starting from the character that is currently marked in the internal buffer.
PushbackReader.unread (int)	Replaces the marked character in the internal buffer with the given character.
PushbackReader.unread (char[], int, int)	Replaces the characters read into the internal buffer from the underlying stream with the given array of characters, starting at the offset given, for the length specified. The characters are replaced starting from the character that is currently marked in the internal buffer.

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

PushbackReader.unread Method (Char[ ])

Replaces the characters read into the internal buffer from the underlying stream with the given array of characters. The characters are replaced starting from the character that is currently marked in the internal buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void unread(
    char[] b) throws java.io.IOException;
```

Parameters

b

The array of characters to replace the characters currently in the internal buffer, starting from the position that is currently marked.

Example

The following example demonstrates how to "push back" the data read from a PushbackReader object.

```
// pushbackreader_unread.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PushbackReader object with an
            // underlying StringReader buffer.
            String s = "This is the internal buffer of StringReader.";
            StringReader stringReader = new StringReader(s);
            PushbackReader pbReader = new PushbackReader(stringReader, 100);

            // Read from the underlying StringReader buffer.
            char[] arr = new char[s.length()];
            if (pbReader.ready())
            {
                pbReader.read(arr, 0, arr.length);
            }
            System.out.println(arr);

            // "Unread" some of the data read from the underlying
            // StringReader buffer.
            String newS = "Replacement internal buffer.";
            char[] newSArr = newS.toCharArray();
            pbReader.unread(newSArr);

            // Close the PushbackReader object and the underlying
            // StringReader object.
            stringReader.close();
            pbReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
This is the internal buffer of StringReader.
*/
```

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

PushbackReader.unread Method (Int32)

Replaces the marked character in the internal buffer with the given character.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void unread(
    int b) throws java.io.IOException;
```

Parameters

b

The character to replaces the last character read into the internal buffer from the underlying stream.

Example

The following example demonstrates how to "push back" the data read from a PushbackReader object.

```
// pushbackreader_unread_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PushbackReader object with an
            // underlying StringReader buffer.
            String s = "This is the internal buffer of StringReader.";
            StringReader stringReader = new StringReader(s);
            PushbackReader pbReader = new PushbackReader(stringReader, 100);

            // Read from the underlying StringReader buffer.
            char[] arr = new char[s.length()];
            if (pbReader.ready())
            {
                pbReader.read(arr, 0, arr.length);
            }
            System.out.println(arr);

            // "Unread" some of the data read from the underlying
            // StringReader buffer.
            String newS = "Replacement internal buffer.";
            for (int i = 0; i < newS.length(); i++)
            {
                pbReader.unread(newS.charAt(i));
            }

            // Close the PushbackReader object and the underlying
            // StringReader object.
            stringReader.close();
            pbReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
```

```
Output:
This is the internal buffer of StringReader.
*/
```

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

PushbackReader.unread Method (Char[ ], Int32, Int32)

Replaces the characters read into the internal buffer from the underlying stream with the given array of characters, starting at the offset given, for the length specified. The characters are replaced starting from the character that is currently marked in the internal buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void unread(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

The array of characters to replace the characters currently in the internal buffer, starting from the position that is currently marked.

off

An offset into the above array of characters. This value represents the index of the first character in the array to be written. This value must be greater than zero.

len

The number of characters to be replaced. This value plus the offset must be less than the overall length of the array.

Example

The following example demonstrates how to "push back" the data read from a PushbackReader object.

```
// pushbackreader_unread_3.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new instance of a PushbackReader object with an
            // underlying StringReader buffer.
            String s = "This is the internal buffer of StringReader.";
            StringReader stringReader = new StringReader(s);
            PushbackReader pbReader = new PushbackReader(stringReader, 100);

            // Read from the underlying StringReader buffer.
            char[] arr = new char[s.length()];
            if (pbReader.ready())
            {
                pbReader.read(arr, 0, arr.length);
            }
            System.out.println(arr);

            // "Unread" some of the data read from the underlying
            // StringReader buffer.
            String newS = "Replacement internal buffer.";
            char[] newSArr = newS.toCharArray();
            pbReader.unread(newSArr, 0, newSArr.length);

            // Close the PushbackReader object and the underlying
```

```
            // StringReader object.
            stringReader.close();
            pbReader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
This is the internal buffer of StringReader.
*/
```

See Also

Reference

PushbackReader Class

Concepts

PushbackReader Members

java.io Package

RandomAccessFile Class

Represents a file used to store data that can be accessed in a non-sequential manner.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.RandomAccessFile
    extends java.lang.Object
    implements java.io.DataOutput, java.io.DataInput
```

Example

The following example demonstrates the close, readDouble, readInt, seek, skipBytes, writeBoolean, writeDouble, and writeInt methods of the RandomAccessFile class.

```
// randomaccessfile_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write some data to the file.
            raf.writeBoolean(true);
            raf.writeDouble(3.14);
            raf.writeInt(2147483647);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Skip over 1 byte (the length of a boolean).
            raf.skipBytes(1);

            // Read in the next double from the file.
            double d = raf.readDouble();
            System.out.println("The double at position 1 is " + d);

            // Read in the next int from the file.
            int i = raf.readInt();
            System.out.println("The int after the double is " + i);

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The double at position 1 is 3.14
The int after the double is 2147483647
```

```
*/
```

Inheritance Hierarchy

java.lang.Object

java.io.RandomAccessFile

See Also

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile Members

Represents a file used to store data that can be accessed in a non-sequential manner.

The following tables list the members exposed by the RandomAccessFile type.

Public Constructors

Name	Description
RandomAccessFile	Overloaded. Initializes a new instance of a RandomAccessFile object.

Public Methods

Name	Description
close	Closes the underlying input stream.
equals	(inherited from Object)
getFD	Gets the file descriptor for the file being accessed.
getFilePointer	Gets the position in the file where the next read or write operation will occur.
hashCode	(inherited from Object)
getClass	(inherited from Object)
length	Gets the length of the file being accessed.
clone	Creates an instance of a RandomAccessFile object that is a shallow copy of the current RandomAccessFile object.
read	Overloaded. Reads the next character or group of characters from the file.
readBoolean	Reads the next byte from the file, interpreting the data as a boolean.
readByte	Reads the next byte from the file.
readChar	Reads the next two bytes from the file, interpreting the data as a char.
readDouble	Reads the next eight bytes from the file, interpreting the data as a double.
readFloat	Reads the next four bytes from the file, interpreting the data as a float.
readFully	Overloaded. Reads the specified number of bytes from the file.
readInt	Reads the next four bytes from the file, interpreting the data as an int.
readLine	Reads the next line from the file.
readLong	Reads the next eight bytes from the file, interpreting the data as a long.
readShort	Reads the next two bytes from the file, interpreting the data as a short.
readUnsignedByte	Reads the next byte from the file, interpreting the data as a ubyte.

readUnsignedShort	Reads the next two bytes from the file, interpreting the data as an unsigned short.
readUTF	Reads a string from the file as UTF (Unicode Transfer Format).
seek	Moves the pointer for the next read or write operation to the given position in the file.
skipBytes	Skips over the next n bytes in the file.
toString	Overridden. Displays a human-readable representation of a RandomAccessFile object.
write	Overloaded. Writes data to the file.
writeBoolean	Writes a boolean to the file.
writeByte	Writes a byte to the file.
writeBytes	Writes a string to the file.
writeChar	Writes a char to the file.
writeChars	Writes a string to the file.
writeDouble	Writes a double to the file.
writeFloat	Writes a float to the file.
writeInt	Writes an int to the file.
writeLong	Writes a long to the file.
writeShort	Writes a short to the file.
writeUTF	Writes a UTF (Unicode Transfer Format) string to the file.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

RandomAccessFile Class

Concepts

java.io Package

RandomAccessFile Constructor

Initializes a new instance of a RandomAccessFile object.

Overload List

Name	Description
RandomAccessFile (File, String)	Initializes a new instance of a RandomAccessFile object using the given file.
RandomAccessFile (String, String)	Initializes a new instance of a RandomAccessFile object using the given file.

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile Constructor (File, String)

Initializes a new instance of a RandomAccessFile object using the given file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.RandomAccessFile(
    java.io.File f,
    java.lang.String mode) throws java.io.IOException;
```

Parameters

f

An instance of a File object representing the file being accessed.

mode

The mode of the file. Either "rw" for read-write access or "r" for read-only access.

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile Constructor (String, String)

Initializes a new instance of a RandomAccessFile object using the given file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.RandomAccessFile(
    java.lang.String filename,
    java.lang.String mode) throws java.io.IOException;
```

Parameters

filename

The name of the file being accessed.

mode

The mode of the file. Either "rw" for read-write access or "r" for read-only access.

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile Methods

Public Methods

Name	Description
close	Closes the underlying input stream.
equals	(inherited from Object)
getFD	Gets the file descriptor for the file being accessed.
getFilePointer	Gets the position in the file where the next read or write operation will occur.
hashCode	(inherited from Object)
getClass	(inherited from Object)
length	Gets the length of the file being accessed.
clone	Creates an instance of a RandomAccessFile object that is a shallow copy of the current RandomAccessFile object.
read	Overloaded. Reads the next character or group of characters from the file.
readBoolean	Reads the next byte from the file, interpreting the data as a boolean.
readByte	Reads the next byte from the file.
readChar	Reads the next two bytes from the file, interpreting the data as a char.
readDouble	Reads the next eight bytes from the file, interpreting the data as a double.
readFloat	Reads the next four bytes from the file, interpreting the data as a float.
readFully	Overloaded. Reads the specified number of bytes from the file.
readInt	Reads the next four bytes from the file, interpreting the data as an int.
readLine	Reads the next line from the file.
readLong	Reads the next eight bytes from the file, interpreting the data as a long.
readShort	Reads the next two bytes from the file, interpreting the data as a short.
readUnsignedByte	Reads the next byte from the file, interpreting the data as a ubyte.
readUnsignedShort	Reads the next two bytes from the file, interpreting the data as an unsigned short.
readUTF	Reads a string from the file as UTF (Unicode Transfer Format).
seek	Moves the pointer for the next read or write operation to the given position in the file.

skipBytes	Skips over the next n bytes in the file.
toString	Overridden. Displays a human-readable representation of a RandomAccessFile object.
write	Overloaded. Writes data to the file.
writeBoolean	Writes a boolean to the file.
writeByte	Writes a byte to the file.
writeBytes	Writes a string to the file.
writeChar	Writes a char to the file.
writeChars	Writes a string to the file.
writeDouble	Writes a double to the file.
writeFloat	Writes a float to the file.
writeInt	Writes an int to the file.
writeLong	Writes a long to the file.
writeShort	Writes a short to the file.
writeUTF	Writes a UTF (Unicode Transfer Format) string to the file.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

RandomAccessFile Class

Concepts

java.io Package

RandomAccessFile.clone Method

Creates an instance of a RandomAccessFile object that is a shallow copy of the current RandomAccessFile object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

Return Value

An instance of a RandomAccessFile object that is a shallow copy of the current RandomAccessFile object.

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.close Method

Closes the underlying input stream.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close a RandomAccessFile object.

```
// randomaccessfile_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Do something useful with the file here.

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.getFD Method

Gets the file descriptor for the file being accessed.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final java.io.FileDescriptor getFD() throws java.io.IOException;
```

Return Value

The file descriptor, represented by a FileDescriptor object, for the file being accessed.

Example

The following example shows how to obtain the FileDescriptor object for a random access file.

```
// randomaccessfile_getfd.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Get the file descriptor for this file.
            FileDescriptor fd = raf.getFD();
            System.out.println("Is the file descriptor valid? " +
                fd.valid());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Is the file descriptor valid? true
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.getFilePointer Method

Gets the position in the file where the next read or write operation will occur.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long getFilePointer() throws java.io.IOException;
```

Return Value

The position in the file where the next read or write operation will occur.

Example

The following example shows how to determine the location within the file where the next read or write will occur.

```
// randomaccessfile_getfilepointer.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write some data to the file.
            raf.writeBoolean(true);
            raf.writeDouble(3.14);
            raf.writeInt(2147483647);

            // Get the location within the file for the next
            // read or write operation.
            long ptr = raf.getFilePointer();
            System.out.println("The next read or write will occur " +
                "at position " + ptr);

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The next read or write will occur at position 13
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.length Method

Gets the length of the file being accessed.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long length() throws java.io.IOException;
```

Return Value

The length of the file being accessed.

Example

The following example displays the length of a random access file.

```
// randomaccessfile_length.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write some data to the file.
            raf.writeBoolean(true);
            raf.writeDouble(3.14);
            raf.writeInt(2147483647);

            // Get the length of the file.
            long len = raf.length();
            System.out.println("The file contains " + len +
                " bytes.");

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The file contains 13 bytes.
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.read Method

Reads the next character or group of characters from the file.

Overload List

Name	Description
RandomAccessFile.read ()	Reads the next character from the file.
RandomAccessFile.read (byte[])	Reads from the file and stores the data read in the provided array.
RandomAccessFile.read (byte[], int, int)	Reads from the file for the length specified and stores the data read in the provided array, starting at an offset.

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.read Method ()

Reads the next character from the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

Return Value

The numeric value of the character read from the file, or -1 if there are no more characters to read.

Example

The following example shows how to read a single character from a random access file.

```
// randomaccessfile_read.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a single character.
            char c = 'a';
            raf.write(c);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next character.
            System.out.println("char: " + (char)raf.read());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
char: a
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.read Method (SByte[])

Reads from the file and stores the data read in the provided array.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] b) throws java.io.IOException;
```

Parameters

b

An array to store the data read from the file.

Return Value

The number of characters read.

Example

The following example shows how to read an array of bytes from a random access file.

```
// randomaccessfile_read_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a byte array.
            String s = "byte array";
            byte[] arr = s.getBytes();
            raf.write(arr);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next byte array.
            byte[] newArr = new byte[arr.length];
            raf.read(newArr);
            System.out.println("byte[]: " + new String(newArr));

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
byte[]: byte array
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.read Method (SByte[], Int32, Int32)

Reads from the file for the length specified and stores the data read in the provided array, starting at an offset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] b,
    int offset,
    int len) throws java.io.IOException;
```

Parameters

b

An array to store the data read from the file.

offset

An offset into the array. This value represents the index of the first character in the array to be written. This value must be greater than zero.

len

The number of characters to be read from the file. This value plus the offset must be less than the overall length of the array.

Return Value

The number of characters read.

Example

The following example shows how to read an array of bytes from a random access file.

```
// randomaccessfile_read_3.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a byte array from position 0 to 6.
            String s = "another byte array";
            byte[] arr2 = s.getBytes();
            raf.write(arr2, 0, 7);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next byte array.
            byte[] newArr2 = new byte[arr2.length];
            raf.read(newArr2, 0, 7);
            System.out.println("byte[]: "+ new String(newArr2));

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
```

```
            {
                System.out.println(ex.toString());
            }
        }
    }

    /*
    Output:
    byte[]: another
    */
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readBoolean Method

Reads the next byte from the file, interpreting the data as a boolean.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final boolean readBoolean() throws java.io.IOException;
```

Return Value

The value of the data read as a boolean.

Example

The following example shows how to read a boolean from a random access file.

```
// randomaccessfile_readboolean.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a boolean.
            raf.writeBoolean(true);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next boolean.
            System.out.println("boolean: " + raf.readBoolean());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
boolean: true
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readByte Method

Reads the next byte from the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final byte readByte() throws java.io.IOException;
```

Return Value

The next byte read from the file.

Example

The following example shows how to read a byte from a random access file.

```
// randomaccessfile_readbyte.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a byte.
            raf.writeByte(22);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next byte.
            System.out.println("byte: " + raf.readByte());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
byte: 22
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readChar Method

Reads the next two bytes from the file, interpreting the data as a char.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final char readChar() throws java.io.IOException;
```

Return Value

The value of the data read as a char.

Example

The following example shows how to read a char from a random access file.

```
// randomaccessfile_readchar.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a char.
            raf.writeChar('Z');

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next char.
            System.out.println("char: " + raf.readChar());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
char: Z
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readDouble Method

Reads the next eight bytes from the file, interpreting the data as a double.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final double readDouble() throws java.io.IOException;
```

Return Value

The value of the data read as a double.

Example

The following example shows how to read a double from a random access file.

```
// randomaccessfile_readdouble.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a double.
            raf.writeDouble(3.14d);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next double.
            System.out.println("double: " + raf.readDouble());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
double: 3.14
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readFloat Method

Reads the next four bytes from the file, interpreting the data as a float.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final float readFloat() throws java.io.IOException;
```

Return Value

The value of the data read as a float.

Example

The following example shows how to read a float from a random access file.

```
// randomaccessfile_readfloat.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a float.
            raf.writeFloat(1.717f);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next float.
            System.out.println("float: " + raf.readFloat());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
float: 1.717
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readFully Method

Reads the specified number of bytes from the file.

Overload List

Name	Description
RandomAccessFile.readFully (byte[])	Reads the number of bytes specified by the given array from the file and fills the array with the data read.
RandomAccessFile.readFully (byte[], int, int)	Reads len bytes from the file and fills the array with the data read, starting at an offset.

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readFully Method (SByte[])

Reads the number of bytes specified by the given array from the file and fills the array with the data read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void readFully(
    byte[] b) throws java.io.IOException;
```

Parameters

b

An array to store the data read from the file.

Example

The following example reads in the entire contents of a random access file and stores it in an array.

```
// randomaccessfile_readfully.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a string.
            String str = "a string\nanother string";
            raf.writeBytes(str);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the entire file.
            byte[] arr = new byte[str.length()];
            raf.readFully(arr);
            String s = new String(arr);
            System.out.println("readFully as string: " + s);

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
readFully as string: a string
another string
*/
```

See Also

Reference
RandomAccessFile Class
Concepts
RandomAccessFile Members
java.io Package

RandomAccessFile.readFully Method (SByte[ ], Int32, Int32)

Reads len bytes from the file and fills the array with the data read, starting at an offset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void readFully(
    byte[] b,
    int offset,
    int len) throws java.io.IOException;
```

Parameters

b

An array to store the data read from the file.

offset

An offset into the array. This value represents the index of the first character in the array to be written. This value must be greater than zero.

len

The number of characters to be read from the file. This value plus the offset must be less than the overall length of the array.

Example

The following example reads in the entire contents of a random access file and stores it in an array.

```
// randomaccessfile_readfully_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a string.
            String str = "a string\nanother string";
            raf.writeBytes(str);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the entire file.
            byte[] arr = new byte[str.length()];
            raf.readFully(arr, 0, arr.length);
            String s = new String(arr);
            System.out.println("readFully as string: " + s);

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
```

```
}

/*
Output:
readFully as string: a string
another string
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readInt Method

Reads the next four bytes from the file, interpreting the data as an int.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int readInt() throws java.io.IOException;
```

Return Value

The value of the data read as an int.

Example

The following example shows how to read an int from a random access file.

```
// randomaccessfile_readint.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out an int.
            raf.writeInt(2147483647);

             // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next int.
            System.out.println("int: " + raf.readInt());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
int: 2147483647
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readLine Method

Reads the next line from the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String readLine() throws java.io.IOException;
```

Return Value

The line read from the file, or null if no end-of-line marker was found.

Example

The following example shows how to read in the next line from a random access file.

```
// randomaccessfile_readline.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a string.
            String str = "a string\nanother string";
            raf.writeBytes(str);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next line.
            System.out.println("line: " + raf.readLine());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
line: a string
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readLong Method

Reads the next eight bytes from the file, interpreting the data as a long.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final long readLong() throws java.io.IOException;
```

Return Value

The value of the data read as a long.

Example

The following example shows how to read a long from a random access file.

```
// randomaccessfile_readlong.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a long.
            raf.writeLong(10000000000L);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next long.
            System.out.println("long: " + raf.readLong());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
long: 10000000000
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readShort Method

Reads the next two bytes from the file, interpreting the data as a short.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final short readShort() throws java.io.IOException;
```

Return Value

The value of the data read as a short.

Example

The following example shows how to read a short from a random access file.

```
// randomaccessfile_readshort.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a short.
            raf.writeShort(255);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next short.
            System.out.println("short: " + raf.readShort());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
short: 255
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readUnsignedByte Method

Reads the next byte from the file, interpreting the data as a ubyte.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int readUnsignedByte() throws java.io.IOException;
```

Return Value

The value of the data read as a ubyte.

Example

The following example shows how to read a ubyte from a random access file.

```
// randomaccessfile_readunsignedbyte.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a byte.
            raf.writeByte(22);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next byte.
            System.out.println("byte: " + raf.readUnsignedByte());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
byte: 22
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readUnsignedShort Method

Reads the next two bytes from the file, interpreting the data as an unsigned short.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final int readUnsignedShort() throws java.io.IOException;
```

Return Value

The value of the data read as an unsigned short.

Example

The following example shows how to read an unsigned short from a random access file.

```
// randomaccessfile_readunsignedshort.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a short.
            raf.writeShort(255);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next short.
            System.out.println("short: " + raf.readUnsignedShort());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
short: 255
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.readUTF Method

Reads a string from the file as UTF (Unicode Transfer Format).

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String readUTF() throws java.io.IOException;
```

Return Value

The string printed in UTF (Unicode Transfer Format).

Example

The following example shows how to read a UTF string from a random access file.

```
// randomaccessfile_readutf.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a UTF string.
            String utf = "UTF string";
            raf.writeUTF(utf);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next UTF string.
            System.out.println("UTF: " + raf.readUTF());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
UTF: UTF string
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.seek Method

Moves the pointer for the next read or write operation to the given position in the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void seek(
    long pos) throws java.io.IOException;
```

Parameters

pos

The position in the file where the next read or write operation will occur.

Example

The following example demonstrates how to position the pointer for the next read or write operation.

```
// randomaccessfile_seek.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write some data to the file.
            raf.writeBoolean(true);
            raf.writeDouble(3.14);
            raf.writeInt(2147483647);

            // Reposition the file pointer to position 5.
            raf.seek(5);

            // Read in the next char from the file.
            char c = raf.readChar();
            System.out.println("The char at position 5 is " + c);

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The char at position 5 is ?
*/
```

See Also

Reference

RandomAccessFile Class

Concepts
RandomAccessFile Members
java.io Package

RandomAccessFile.skipBytes Method

Skips over the next n bytes in the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int skipBytes(
    int n) throws java.io.IOException;
```

Parameters

n

The number of bytes to skip over before the next read or write operation.

Return Value

The actual number of bytes skipped over. This can be less than n if there are less than n bytes remaining in the file when skipBytes is called.

Example

The following example shows how to skip over data in a random access file.

```
// randomaccessfile_skipbytes.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write some data to the file.
            raf.writeBoolean(true);
            raf.writeDouble(3.14);
            raf.writeInt(2147483647);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Skip over 1 byte (the length of a boolean).
            raf.skipBytes(1);

            // Read in the next double from the file.
            double d = raf.readDouble();
            System.out.println("The double at position 1 is " + d);

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
```

```
Output:
The double at position 1 is 3.14
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.toString Method

Displays a human-readable representation of a RandomAccessFile object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

Return Value

A human-readable representation of a RandomAccessFile object.

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.write Method

Writes data to the file.

Overload List

Name	Description
RandomAccessFile.write (int)	Writes the next character to the file.
RandomAccessFile.write (byte[])	Writes the data contained in the given array to the file.
RandomAccessFile.write (byte[], int, int)	Writes len characters from the given array, starting at an offset, to the file.

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.write Method (Int32)

Writes the next character to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int b) throws java.io.IOException;
```

Parameters

b

The character to write to the file.

Example

The following example shows how to write a single character to a random access file.

```
// randomaccessfile_write.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a single character.
            char c = 'a';
            raf.write(c);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next character.
            System.out.println("char: " + (char)raf.read());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
char: a
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

RandomAccessFile.write Method (SByte[ ])

Writes the data contained in the given array to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] b) throws java.io.IOException;
```

Parameters

b

An array containing the data to write to the file.

Example

The following example shows how to write an array of bytes to a random access file.

```
// randomaccessfile_write_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a byte array.
            String s = "byte array";
            byte[] arr = s.getBytes();
            raf.write(arr);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next byte array.
            byte[] newArr = new byte[arr.length];
            raf.read(newArr);
            System.out.println("byte[]: " + new String(newArr));

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
byte[]: byte array
*/
```

See Also

Reference

RandomAccessFile Class
Concepts
RandomAccessFile Members
java.io Package

RandomAccessFile.write Method (SByte[ ], Int32, Int32)

Writes len characters from the given array, starting at an offset, to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    byte[] b,
    int offset,
    int len) throws java.io.IOException;
```

Parameters

b

An array containing the data to write to the file.

offset

An offset into the array. This value represents the index of the first character in the array to be written to the file. This value must be greater than zero.

len

The number of characters to be written to the file starting from the offset. This value plus the offset must be less than the overall length of the array.

Example

The following example shows how to write an array of bytes to a random access file.

```
// randomaccessfile_write_3.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a byte array from position 0 to 6.
            String s = "another byte array";
            byte[] arr2 = s.getBytes();
            raf.write(arr2, 0, 7);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next byte array.
            byte[] newArr2 = new byte[arr2.length];
            raf.read(newArr2, 0, 7);
            System.out.println("byte[]: "+ new String(newArr2));

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
```

```
        }
    }
}

/*
Output:
byte[]: another
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.writeBoolean Method

Writes a boolean to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeBoolean(
    boolean bool) throws java.io.IOException;
```

Parameters

bool

The boolean to write to the file.

Example

The following example shows how to write a boolean to a random access file.

```
// randomaccessfile_writeboolean.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a boolean.
            raf.writeBoolean(true);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next boolean.
            System.out.println("boolean: " + raf.readBoolean());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
boolean: true
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.writeByte Method

Writes a byte to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeByte(
    int b) throws java.io.IOException;
```

Parameters

b

The byte to write to the file.

Example

The following example shows how to write a byte to a random access file.

```
// randomaccessfile_writebyte.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a byte.
            raf.writeByte(22);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next byte.
            System.out.println("byte: " + raf.readByte());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
byte: 22
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.writeBytes Method

Writes a string to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeBytes(
    java.lang.String b) throws java.io.IOException;
```

Parameters

b

The string to write to the file.

Example

The following example shows how to write a string to a random access file.

```
// randomaccessfile_writebytes.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a string.
            String str = "a string";
            raf.writeBytes(str);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the string.
            byte[] arr = new byte[str.length()];
            raf.read(arr);
            String s = new String(arr);
            System.out.println("bytes: "  + s);

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
bytes: a string
*/
```

See Also

Reference

RandomAccessFile Class
Concepts
RandomAccessFile Members
java.io Package

RandomAccessFile.writeChar Method

Writes a char to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeChar(
    int b) throws java.io.IOException;
```

Parameters

b

The char to write to the file.

Example

The following example shows how to write a char to a random access file.

```
// randomaccessfile_writechar.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a char.
            raf.writeChar('Z');

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next char.
            System.out.println("char: " + raf.readChar());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
char: Z
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.writeChars Method

Writes a string to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeChars(
    java.lang.String s) throws java.io.IOException;
```

Parameters

s

The string to write to the file.

Example

The following example shows how to write a string to a random access file.

```
// randomaccessfile_writechars.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a string.
            String str = "a string";
            raf.writeChars(str);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the string.
            byte[] arr = new byte[str.length() * 2];
            raf.read(arr);
            String s = new String(arr);
            System.out.println("chars: "  + s);

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
chars:  a   s t r i n g
*/
```

See Also

Reference

RandomAccessFile Class
Concepts
RandomAccessFile Members
java.io Package

RandomAccessFile.writeDouble Method

Writes a double to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeDouble(
    double d) throws java.io.IOException;
```

Parameters

d

The double to write to the file.

Example

The following example shows how to write a double to a random access file.

```
// randomaccessfile_writedouble.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a double.
            raf.writeDouble(3.14d);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next double.
            System.out.println("double: " + raf.readDouble());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
double: 3.14
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.writeFloat Method

Writes a float to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeFloat(
    float f) throws java.io.IOException;
```

Parameters

f

The float to write to the file.

Example

The following example shows how to write a float to a random access file.

```
// randomaccessfile_writefloat.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a float.
            raf.writeFloat(1.717f);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next float.
            System.out.println("float: " + raf.readFloat());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
float: 1.717
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.writeInt Method

Writes an int to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeInt(
    int i) throws java.io.IOException;
```

Parameters

i

The int to write to the file.

Example

The following example shows how to write an int to a random access file.

```
// randomaccessfile_writeint.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out an int.
            raf.writeInt(2147483647);

             // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next int.
            System.out.println("int: " + raf.readInt());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
int: 2147483647
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.writeLong Method

Writes a long to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeLong(
    long l) throws java.io.IOException;
```

Parameters

l

The long to write to the file.

Example

The following example shows how to write a long to a random access file.

```
// randomaccessfile_writelong.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a long.
            raf.writeLong(10000000000L);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next long.
            System.out.println("long: " + raf.readLong());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
long: 10000000000
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.writeShort Method

Writes a short to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeShort(
    int s) throws java.io.IOException;
```

Parameters

s

The short to write to the file.

Example

The following example shows how to write a short to a random access file.

```
// randomaccessfile_writeshort.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a short.
            raf.writeShort(255);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next short.
            System.out.println("short: " + raf.readShort());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
short: 255
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

java.io Package

RandomAccessFile.writeUTF Method

Writes a UTF (Unicode Transfer Format) string to the file.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public final void writeUTF(
    java.lang.String s) throws java.io.IOException;
```

Parameters

s

The UTF string to write to the file.

Example

The following example shows how to write a UTF string to a random access file.

```
// randomaccessfile_writeutf.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new random access file.
            RandomAccessFile raf = new RandomAccessFile(
                "C:\\RandomFile.txt", "rw");

            // Write out a UTF string.
            String utf = "UTF string";
            raf.writeUTF(utf);

            // Reposition the file pointer to position 0.
            raf.seek(0);

            // Read in the next UTF string.
            System.out.println("UTF: " + raf.readUTF());

            // Close the file.
            raf.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
UTF: UTF string
*/
```

See Also

Reference

RandomAccessFile Class

Concepts

RandomAccessFile Members

Reader Class

An abstract class representing the base for all character readers in the java.io package.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.io.Reader
    extends java.lang.Object
```

Example

The following example demonstrates the close, mark, markSupported, read, and reset methods of the StringReader class, which is a specialization of the abstract Reader class.

```
// reader_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the string.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the StringReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
The next letter read after reset is: e
*/
```

Inheritance Hierarchy

java.lang.Object

java.io.Reader

Derived Classes

See Also

Concepts

Reader Members

java.io Package

Reader Members

An abstract class representing the base for all character readers in the java.io package.

The following tables list the members exposed by the Reader type.

Public Constructors

Name	Description
Reader	Overloaded. Initializes a new instance of a Reader object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader.

Public Methods

Name	Description
close	An abstract method that must be implemented by subclasses. This method should take care of releasing any internal buffers and resources when called.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Sets the mark to the next character of the characters to be read.
markSupported	Determines whether the current position of the character can be marked.
clone	Creates a shallow copy of the object.
read	Overloaded. Reads the next character or characters of the characters to be read.
ready	Determines whether the reader is ready for reading.
reset	Resets the reader for the characters such that the next character read is the character that is marked.
skip	Skips over the next count characters of the characters being read or until the end of the array, whichever is less.
toString	Overridden. Displays a human readable representation of the reader.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Reader Class

Concepts

java.io Package

Reader Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader.

See Also

Reference

Reader Class

Concepts

java.io Package

Reader.lock Field

The object used to synchronize operations on the reader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Object lock;
```

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader Constructor

Initializes a new instance of a Reader object.

Overload List

Name	Description
Reader ()	Initializes a new instance of a Reader object.
Reader (Object)	Initializes a new instance of a Reader object. The given lock is used to synchronize operations on the reader.

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader Constructor ()

Initializes a new instance of a Reader object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.Reader();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader Constructor (Object)

Initializes a new instance of a Reader object. The given lock is used to synchronize operations on the reader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.Reader(
    java.lang.Object lock);
```

Parameters

lock

The object used to synchronize operations on the reader.

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader Methods

Public Methods

Name	Description
close	An abstract method that must be implemented by subclasses. This method should take care of releasing any internal buffers and resources when called.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Sets the mark to the next character of the characters to be read.
markSupported	Determines whether the current position of the character can be marked.
clone	Creates a shallow copy of the object.
read	Overloaded. Reads the next character or characters of the characters to be read.
ready	Determines whether the reader is ready for reading.
reset	Resets the reader for the characters such that the next character read is the character that is marked.
skip	Skips over the next count characters of the characters being read or until the end of the array, whichever is less.
toString	Overridden. Displays a human readable representation of the reader.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Reader Class

Concepts

java.io Package

Reader.clone Method

Creates a shallow copy of the object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

Return Value

A shallow copy of the object.

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.close Method

An abstract method that must be implemented by subclasses. This method should take care of releasing any internal buffers and resources when called.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close a StringReader object, which is a specialization of the abstract Reader class.

```
// reader_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying String.
            char[] newArr = new char[s.length()];
            reader.read(newArr);
            System.out.println("The new array contains: " +
                newArr);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The new array contains: Hello
*/
```

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.mark Method

Sets the mark to the next character of the characters to be read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void mark(
    int readLimit) throws java.io.IOException;
```

Parameters

readLimit

Derived classes use this parameter differently. Consult the documentation for the particular reader that you are using.

Example

The following example demonstrates the effect that the mark method has when you call reset. The StringReader class is a specialization of the abstract Reader class.

```
// reader_mark.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the string.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the StringReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
```

```
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The next letter read after reset is: e
*/
```

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.markSupported Method

Determines whether the current position of the character can be marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

Return Value

true if the current position of the character can be marked; false otherwise.

Example

The following example demonstrates the effect that the mark method has when you call reset. The StringReader class is a specialization of the abstract Reader class.

```
// reader_marksupported.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the string.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the StringReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }
}

/*
Output:
The next letter read after reset is: e
*/
```

Remarks

The default implementation for markSupported always returns false. Subclasses should return true if they implement functionality for supporting mark and reset.

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.read Method

Reads the next character or characters of the characters to be read.

Overload List

Name	Description
Reader.read ()	Reads the next character of the characters to be read.
Reader.read (char[])	Reads the characters in the array starting at an offset for the length specified and stores them in the provided array of characters.
Reader.read (char[], int, int)	Reads the characters starting at an offset for the length specified, and stores them in the provided array of characters.

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.read Method ()

Reads the next character of the characters to be read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

Return Value

A numeric value of the character that was read. Derived classes use this return value differently. Consult the documentation for the particular reader that you are using.

Example

The following example demonstrates how to read one character at a time from the StringReader buffer, which is a specialization of the abstract Reader class.

```
// reader_read.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string.
            for (int i = 0; i < s.length(); i++)
            {
                if (reader.ready())
                {
                    char charRead = (char)reader.read();
                    System.out.println(charRead);
                }
                else
                {
                    break;
                }
            }

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
H
e
l
l
```

```
o
*/
```

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.read Method (Char[])

Reads the characters in the array starting at an offset for the length specified and stores them in the provided array of characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    char[] buf) throws java.io.IOException;
```

Parameters

buf

The array of characters to store the characters read.

Return Value

The number of characters read.

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.read Method (Char[ ], Int32, Int32)

Reads the characters starting at an offset for the length specified, and stores them in the provided array of characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract int read(
    char[] buf,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buf

The array of characters to store the characters read.

offset

An offset into the characters. This value represents the index of the first character to be read. This value must be greater than zero.

count

The number of characters to be read starting from the offset. This value plus the offset must be less than the overall length of the characters.

Return Value

The number of characters read.

Example

The following example demonstrates how to read from the StringReader buffer and populate an already existing array with the contents just read. The StringReader class is a specialization of the abstract Reader class.

```
// reader_read_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying array.
            char[] newArr = new char[s.length() + 6];
            newArr[6] = 'W';
            newArr[7] = 'o';
            newArr[8] = 'r';
            newArr[9] = 'l';
            newArr[10] = 'd';

            reader.read(newArr, 0, s.length());
            System.out.println("The new array contains: " +
                newArr);

            // Close the StringReader.
            reader.close();
        }
```

```
            catch (IOException ex)
            {
                System.out.println(ex.toString());
            }
        }
    }

    /*
    Output:
    The new array contains: Hello World
    */
```

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.ready Method

Determines whether the reader is ready for reading.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean ready() throws java.io.IOException;
```

Return Value

true if the characters are ready for reading; false otherwise.

Example

The following example demonstrates how to determine if a StringReader object is ready for reading. The StringReader class is a specialization of the abstract Reader class.

```
// reader_ready.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string.
            for (int i = 0; i < s.length(); i++)
            {
                if (reader.ready())
                {
                    char charRead = (char)reader.read();
                    System.out.println(charRead);
                }
                else
                {
                    break;
                }
            }

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
H
e
l
l
o
```

```
*/
```

Remarks

The default implementation for ready always returns false. Subclasses should override this method and return the appropriate boolean value.

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.reset Method

Resets the reader for the characters such that the next character read is the character that is marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void reset() throws java.io.IOException;
```

Example

The following example demonstrates the effect that the mark method has when you call reset. The StringReader class is a specialization of the abstract Reader class.

```
// reader_reset.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the string.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the StringReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
```

```
Output:
The next letter read after reset is: e
*/
```

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.skip Method

Skips over the next count characters of the characters being read or until the end of the array, whichever is less.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long count) throws java.io.IOException;
```

Parameters

count

The number of characters to skip over. This value must be greater than zero. If this number is greater than the number of characters remaining, then only the number of remaining characters will be skipped.

Return Value

The number of characters skipped.

Example

The following example demonstrates how to skip over character while you are performing a read operation on a StringReader object, which is a specialization of the abstract Reader class.

```
// reader_skip.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string.
            for (int i = 0; i < s.length(); i++)
            {
                if (reader.ready())
                {
                    char charRead = (char)reader.read();
                    System.out.println(charRead);

                    // Skip over one character.
                    long num = reader.skip(1);
                    System.out.println("skipping " + num);
                }
                else
                {
                    break;
                }
            }

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }
}

/*
Output:
H
skipping 1
l
skipping 1
o
skipping 0
?
skipping 0
?
skipping 0
*/
```

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

Reader.toString Method

Displays a human readable representation of the reader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

Return Value

A human readable representation of the reader.

See Also

Reference

Reader Class

Concepts

Reader Members

java.io Package

SequenceInputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.SequenceInputStream
    extends java.io.InputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

   java.io.SequenceInputStream

See Also

Concepts

SequenceInputStream Members

java.io Package

SequenceInputStream Members

The following tables list the members exposed by the SequenceInputStream type.

Public Constructors

Name	Description
SequenceInputStream	Overloaded.

Public Methods

Name	Description
available	Overridden.
close	Overridden.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	(inherited from InputStream)
skip	(inherited from InputStream)
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SequenceInputStream Class

Concepts

java.io Package

SequenceInputStream Constructor

Overload List

Name	Description
SequenceInputStream (Enumeration)	
SequenceInputStream (InputStream, InputStream)	

See Also

Reference

SequenceInputStream Class

Concepts

SequenceInputStream Members

java.io Package

SequenceInputStream Constructor (Enumeration)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.SequenceInputStream(
    java.util.Enumeration e);
```

Parameters

e

See Also

Reference

SequenceInputStream Class

Concepts

SequenceInputStream Members

java.io Package

SequenceInputStream Constructor (InputStream, InputStream)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.SequenceInputStream(
    java.io.InputStream s1,
    java.io.InputStream s2);
```

Parameters

s1

s2

See Also

Reference

SequenceInputStream Class

Concepts

SequenceInputStream Members

java.io Package

SequenceInputStream Methods

Public Methods

Name	Description
available	Overridden.
close	Overridden.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	(inherited from InputStream)
skip	(inherited from InputStream)
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SequenceInputStream Class

Concepts

java.io Package

SequenceInputStream.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int available() throws java.io.IOException;
```

See Also

Reference

SequenceInputStream Class

Concepts

SequenceInputStream Members

java.io Package

SequenceInputStream.close Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close() throws java.io.IOException;
```

See Also

Reference

SequenceInputStream Class

Concepts

SequenceInputStream Members

java.io Package

SequenceInputStream.read Method

Overload List

Name	Description
SequenceInputStream.read ()	
SequenceInputStream.read (byte[])	
SequenceInputStream.read (byte[], int, int)	

See Also

Reference

SequenceInputStream Class

Concepts

SequenceInputStream Members

java.io Package

SequenceInputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

See Also

Reference

SequenceInputStream Class

Concepts

SequenceInputStream Members

java.io Package

SequenceInputStream.read Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    byte[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

off

len

See Also

Reference

SequenceInputStream Class

Concepts

SequenceInputStream Members

java.io Package

StreamCorruptedException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.StreamCorruptedException
    extends java.io.ObjectStreamException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.ObjectStreamException

java.io.StreamCorruptedException

See Also

Concepts

StreamCorruptedException Members

java.io Package

StreamCorruptedException Members

The following tables list the members exposed by the StreamCorruptedException type.

Public Constructors

Name	Description
StreamCorruptedException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StreamCorruptedException Class

Concepts

java.io Package

StreamCorruptedException Constructor

Overload List

Name	Description
StreamCorruptedException ()	
StreamCorruptedException (String)	
StreamCorruptedException (SerializationInfo, StreamingContext)	
StreamCorruptedException (String, Exception)	

See Also

Reference

StreamCorruptedException Class

Concepts

StreamCorruptedException Members

java.io Package

StreamCorruptedException Constructor ()

Initializes a new instance of the StreamCorruptedException Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.StreamCorruptedException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

StreamCorruptedException Class

Concepts

StreamCorruptedException Members

java.io Package

StreamCorruptedException Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.StreamCorruptedException(
    java.lang.String msg);
```

Parameters

msg

See Also

Reference

StreamCorruptedException Class

Concepts

StreamCorruptedException Members

java.io Package

StreamCorruptedException Constructor (SerializationInfo, StreamingContext)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.StreamCorruptedException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

StreamCorruptedException Class

Concepts

StreamCorruptedException Members

java.io Package

StreamCorruptedException Constructor (String, Exception)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.StreamCorruptedException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also

Reference

StreamCorruptedException Class

Concepts

StreamCorruptedException Members

java.io Package

StreamCorruptedException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StreamCorruptedException Class

Concepts

java.io Package

StreamCorruptedException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

StreamCorruptedException Class

Concepts

java.io Package

StreamTokenizer Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.StreamTokenizer
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.io.StreamTokenizer

See Also

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer Members

The following tables list the members exposed by the StreamTokenizer type.

Public Constructors

Name	Description
StreamTokenizer	Overloaded.

Public Fields

Name	Description
nval	
sval	
TT_EOF	
TT_EOL	
TT_NUMBER	
TT_WORD	
ttype	

Public Methods

Name	Description
commentChar	
eolIsSignificant	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
lineno	
lowerCaseMode	
clone	
nextToken	
ordinaryChar	
ordinaryChars	
parseNumbers	

pushBack	
quoteChar	
resetSyntax	
slashSlashComments	
slashStarComments	
toString	Overridden.
whitespaceChars	
wordChars	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StreamTokenizer Class

Concepts

java.io Package

StreamTokenizer Fields

Public Fields

Name	Description
nval	
sval	
TT_EOF	
TT_EOL	
TT_NUMBER	
TT_WORD	
ttype	

See Also

Reference

StreamTokenizer Class

Concepts

java.io Package

StreamTokenizer.nval Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public double nval;
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.sval Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String sval;
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.TT_EOF Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final int TT_EOF;
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.TT_EOL Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final int TT_EOL;
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.TT_NUMBER Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final int TT_NUMBER;
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.TT_WORD Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public static final int TT_WORD;
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.ttype Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int ttype;
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer Constructor

Overload List

Name	Description
StreamTokenizer (InputStream)	
StreamTokenizer (Reader)	

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer Constructor (InputStream)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.StreamTokenizer(
    java.io.InputStream in);
```

Parameters

in

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer Constructor (Reader)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.StreamTokenizer(
    java.io.Reader reader);
```

Parameters

reader

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer Methods

Public Methods

Name	Description
commentChar	
eolIsSignificant	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
lineno	
lowerCaseMode	
clone	
nextToken	
ordinaryChar	
ordinaryChars	
parseNumbers	
pushBack	
quoteChar	
resetSyntax	
slashSlashComments	
slashStarComments	
toString	Overridden.
whitespaceChars	
wordChars	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StreamTokenizer Class

Concepts
java.io Package

StreamTokenizer.clone Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.commentChar Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void commentChar(
    int ch);
```

Parameters

ch

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.eolIsSignificant Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void eolIsSignificant(
    boolean flag);
```

Parameters

flag

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.lineno Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int lineno();
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.lowerCaseMode Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void lowerCaseMode(
    boolean flag);
```

Parameters

flag

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.nextToken Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int nextToken() throws java.io.IOException;
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.ordinaryChar Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void ordinaryChar(
    int ch);
```

Parameters

ch

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.ordinaryChars Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void ordinaryChars(
    int low,
    int hi);
```

Parameters

low

hi

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.parseNumbers Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void parseNumbers();
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.pushBack Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void pushBack();
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.quoteChar Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void quoteChar(
    int ch);
```

Parameters

ch

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.resetSyntax Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void resetSyntax();
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.slashSlashComments Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void slashSlashComments(
    boolean flag);
```

Parameters

flag

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.slashStarComments Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void slashStarComments(
    boolean flag);
```

Parameters

flag

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.toString Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.whitespaceChars Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void whitespaceChars(
    int low,
    int hi);
```

Parameters

low

hi

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StreamTokenizer.wordChars Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void wordChars(
    int low,
    int hi);
```

Parameters

low

hi

See Also

Reference

StreamTokenizer Class

Concepts

StreamTokenizer Members

java.io Package

StringBufferInputStream Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.StringBufferInputStream
    extends java.io.InputStream
```

Inheritance Hierarchy

java.lang.Object

  java.io.InputStream

   java.io.StringBufferInputStream

See Also

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream Members

The following tables list the members exposed by the StringBufferInputStream type.

Public Constructors

Name	Description
StringBufferInputStream	

Public Fields

Name	Description
buffer	
count	
pos	

Public Methods

Name	Description
available	Overridden.
close	(inherited from InputStream)
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden.
skip	Overridden.
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringBufferInputStream Class

Concepts

java.io Package

StringBufferInputStream Fields

Public Fields

Name	Description
buffer	
count	
pos	

See Also

Reference

StringBufferInputStream Class

Concepts

java.io Package

StringBufferInputStream.buffer Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.String buffer;
```

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream.count Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int count;
```

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream.pos Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected int pos;
```

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream Constructor

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.StringBufferInputStream(
    java.lang.String s);
```

Parameters

s

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream Methods

Public Methods

Name	Description
available	Overridden.
close	(inherited from InputStream)
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	(inherited from InputStream)
markSupported	(inherited from InputStream)
clone	(inherited from InputStream)
read	Overloaded. Overridden.
reset	Overridden.
skip	Overridden.
toString	(inherited from InputStream)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringBufferInputStream Class

Concepts

java.io Package

StringBufferInputStream.available Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int available();
```

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream.read Method

Overload List

Name	Description
StringBufferInputStream.read ()	
StringBufferInputStream.read (byte[])	
StringBufferInputStream.read (byte[], int, int)	

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream.read Method ()

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int read();
```

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream.read Method (SByte[ ], Int32, Int32)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized int read(
    byte[] b,
    int off,
    int len);
```

Parameters

b

off

len

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream.reset Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized void reset();
```

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringBufferInputStream.skip Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public synchronized long skip(
    long n);
```

Parameters

n

See Also

Reference

StringBufferInputStream Class

Concepts

StringBufferInputStream Members

java.io Package

StringReader Class

Provides an implementation of the abstract Reader class used to read characters from a given string.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.StringReader
    extends java.io.Reader
```

Example

The following example demonstrates the close, mark, markSupported, read, and reset methods of the StringReader class.

```
// stringreader_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the string.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the StringReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
```

```
Output:
The next letter read after reset is: e
*/
```

Inheritance Hierarchy

java.lang.Object

java.io.Reader

java.io.StringReader

See Also

Concepts

StringReader Members

java.io Package

StringReader Members

Provides an implementation of the abstract Reader class used to read characters from a given string.

The following tables list the members exposed by the StringReader type.

Public Constructors

Name	Description
StringReader	Initializes a new instance of a StringReader object with the given string.

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader.(inherited from Reader)

Public Methods

Name	Description
close	Overridden. Closes the reader.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Sets the mark to the next character in the string to be read.
markSupported	Overridden. Determines whether this reader supports marking.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters in the string.
ready	Overridden. Determines whether the reader is ready for reading.
reset	Overridden. Resets the reader for the string such that the next character read is the character that is marked.
skip	Overridden. Skips over the next n characters in the string or until the end of the string, whichever is less.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringReader Class

Concepts

java.io Package

StringReader Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the reader. (inherited from Reader)

See Also

Reference

StringReader Class

Concepts

java.io Package

StringReader Constructor

Initializes a new instance of a StringReader object with the given string.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.StringReader(
    java.lang.String s);
```

Parameters

s

The string to be read.

See Also
Reference
StringReader Class
Concepts
StringReader Members
java.io Package

StringReader Methods

Public Methods

Name	Description
close	Overridden. Closes the reader.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
mark	Overridden. Sets the mark to the next character in the string to be read.
markSupported	Overridden. Determines whether this reader supports marking.
clone	Creates a shallow copy of the object. (inherited from Reader)
read	Overloaded. Overridden. Reads the next character or characters in the string.
ready	Overridden. Determines whether the reader is ready for reading.
reset	Overridden. Resets the reader for the string such that the next character read is the character that is marked.
skip	Overridden. Skips over the next n characters in the string or until the end of the string, whichever is less.
toString	Displays a human readable representation of the reader. (inherited from Reader)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringReader Class

Concepts

java.io Package

StringReader.close Method

Closes the reader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close();
```

Example

The following example demonstrates how to close a StringReader object.

```
// stringreader_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying String.
            char[] newArr = new char[s.length()];
            reader.read(newArr);
            System.out.println("The new array contains: " +
                newArr);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The new array contains: Hello
*/
```

See Also

Reference

StringReader Class

Concepts

StringReader Members

java.io Package

StringReader.mark Method

Sets the mark to the next character in the string to be read.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void mark(
    int readlimit) throws java.io.IOException;
```

Parameters

readlimit

Ignored.

Example

The following example demonstrates the effect that the mark method has when you call reset.

```
// stringreader_mark.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the string.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the StringReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }
}

/*
Output:
The next letter read after reset is: e
*/
```

See Also

Reference

StringReader Class

Concepts

StringReader Members

java.io Package

StringReader.markSupported Method

Determines whether this reader supports marking.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean markSupported();
```

Return Value

Always true.

Example

The following example demonstrates the effect that the mark method has when you call reset.

```
// stringreader_marksupported.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the string.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the StringReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}
```

```
/*
Output:
The next letter read after reset is: e
*/
```

See Also

Reference

StringReader Class

Concepts

StringReader Members

java.io Package

StringReader.read Method

Reads the next character or characters in the string.

Overload List

Name	Description
StringReader.read ()	Reads the next character in the string.
StringReader.read (char[])	Reads the characters in the array starting at an offset for the length specified and stores the m in the provided array of characters.
StringReader.read (char[], int, int)	Reads the characters in the underlying string for the length specified and stores them in the provided array of characters, starting at an offset.

See Also

Reference

StringReader Class

Concepts

StringReader Members

java.io Package

StringReader.read Method ()

Reads the next character in the string.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read() throws java.io.IOException;
```

Return Value

The numeric value of the character read from the current position in string, or -1 if all the characters were already read.

Example

The following example demonstrates how to read one character at a time from the StringReader buffer.

```
// stringreader_read.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string.
            for (int i = 0; i < s.length(); i++)
            {
                if (reader.ready())
                {
                    char charRead = (char)reader.read();
                    System.out.println(charRead);
                }
                else
                {
                    break;
                }
            }

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
H
e
l
l
o
*/
```

See Also

Reference

StringReader Class

Concepts

StringReader Members

java.io Package

StringReader.read Method (Char[ ], Int32, Int32)

Reads the characters in the underlying string for the length specified and stores them in the provided array of characters, starting at an offset.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public int read(
    char[] b,
    int off,
    int len) throws java.io.IOException;
```

Parameters

b

The array of characters to store the characters read.

off

An offset into the array of characters. This value represents the index of the first character in the array to be written. This value must be greater than zero.

len

The number of characters to be read from the underlying string. This value plus the offset must be less than the overall length of the array.

Return Value

The number of characters read.

Example

The following example demonstrates how to read from the StringReader buffer and populate an already existing array with the contents just read.

```
// stringreader_read_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying array.
            char[] newArr = new char[s.length() + 6];
            newArr[6] = 'W';
            newArr[7] = 'o';
            newArr[8] = 'r';
            newArr[9] = 'l';
            newArr[10] = 'd';

            reader.read(newArr, 0, s.length());
            System.out.println("The new array contains: " +
                newArr);

            // Close the StringReader.
            reader.close();
```

```
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The new array contains: Hello World
*/
```

See Also

Reference

StringReader Class

Concepts

StringReader Members

java.io Package

StringReader.ready Method

Determines whether the reader is ready for reading.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public boolean ready();
```

Return Value

Always true.

Example

The following example demonstrates how to determine if a StringReader object is ready for reading.

```
// stringreader_ready.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string.
            for (int i = 0; i < s.length(); i++)
            {
                if (reader.ready())
                {
                    char charRead = (char)reader.read();
                    System.out.println(charRead);
                }
                else
                {
                    break;
                }
            }

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
H
e
l
l
o
*/
```

See Also

Reference

StringReader Class

Concepts

StringReader Members

java.io Package

StringReader.reset Method

Resets the reader for the string such that the next character read is the character that is marked.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void reset() throws java.io.IOException;
```

Example

The following example demonstrates the effect that the mark method has when you call reset.

```
// stringreader_reset.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string one at a time.
            char firstLetter = (char)reader.read();

            // Set the mark.
            if (reader.markSupported())
            {
                // The 0 is ignored.
                reader.mark(0);
            }

            // Continue reading from the string.
            char secondLetter = (char)reader.read();
            char thirdLetter = (char)reader.read();
            char fourthLetter = (char)reader.read();
            char fifthLetter = (char)reader.read();

            // Reset the StringReader.
            reader.reset();

            // Get the first character read after the reset.
            char nextLetter = (char)reader.read();
            System.out.println("The next letter read after reset is: " +
                nextLetter);

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
```

```
The next letter read after reset is: e
*/
```

See Also

Reference

StringReader Class

Concepts

StringReader Members

java.io Package

StringReader.skip Method

Skips over the next n characters in the string or until the end of the string, whichever is less.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public long skip(
    long n) throws java.io.IOException;
```

Parameters

n

The number of characters to skip over. This value must be greater than zero. If this number is greater than the number of characters remaining in the string, then only the number of remaining characters will be skipped.

Return Value

The number of characters skipped.

Example

The following example demonstrates how to skip over character while you are performing a read operation on a StringReader object.

```
// stringreader_skip.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a new StringReader.
            String s = "Hello";
            StringReader reader = new StringReader(s);

            // Read from the underlying string.
            for (int i = 0; i < s.length(); i++)
            {
                if (reader.ready())
                {
                    char charRead = (char)reader.read();
                    System.out.println(charRead);

                    // Skip over one character.
                    long num = reader.skip(1);
                    System.out.println("skipping " + num);
                }
                else
                {
                    break;
                }
            }

            // Close the StringReader.
            reader.close();
        }
        catch (IOException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }
}

/*
Output:
H
skipping 1
l
skipping 1
o
skipping 0
?
skipping 0
?
skipping 0
*/
```

See Also

Reference

StringReader Class

Concepts

StringReader Members

java.io Package

StringWriter Class

Provides an implementation of the abstract Writer class used to write characters into a StringBuffer object that can be resized as required.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.StringWriter
    extends java.io.Writer
```

Example

The following example demonstrates the close, flush, getBuffer, and write methods of the StringWriter class.

```
// stringwriter_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        // Start with the 7th character and write to the end.
        String s = "Hello World";
        writer.write("Hello World", 6, s.length() - 6);

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.toString());

        // Now flush the buffer we just populated.
        writer.flush();

        // Print out the contents of the StringWriter buffer.
        System.out.println("The post-flush StringWriter buffer " +
            "contains: " + writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: World
The post-flush StringWriter buffer contains: World
*/
```

Inheritance Hierarchy

java.lang.Object

  java.io.Writer

   java.io.StringWriter

See Also

Concepts

StringWriter Members

Provides an implementation of the abstract Writer class used to write characters into a StringBuffer object that can be resized as required.

The following tables list the members exposed by the StringWriter type.

Public Constructors

Name	Description
StringWriter	Overloaded. Initializes a new instance of a StringWriter object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer.(inherited from Writer)

Public Methods

Name	Description
close	Overridden. Closes the writer once the string has been processed.
equals	(inherited from Object)
flush	Overridden. Flushes the buffer representing the string to be written.
getBuffer	Retrieves the StringBuffer object used to store the string.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Overridden. Displays a human readable representation of a StringWriter object. The contents of the buffer holding t he string is returned.
write	Overloaded. Appends a character or characters to the end of the string buffer.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringWriter Class

Concepts

java.io Package

StringWriter Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer. (inherited from Writer)

See Also

Reference

StringWriter Class

Concepts

java.io Package

StringWriter Constructor

Initializes a new instance of a StringWriter object.

Overload List

Name	Description
StringWriter ()	Initializes a new instance of a StringWriter object.
StringWriter (int)	Initializes a new instance of a StringWriter object. The buffer containing the string is initialized to the given size.

See Also

Reference

StringWriter Class

Concepts

StringWriter Members

java.io Package

StringWriter Constructor ( )

Initializes a new instance of a StringWriter object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.StringWriter();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

StringWriter Class

Concepts

StringWriter Members

java.io Package

StringWriter Constructor (Int32)

Initializes a new instance of a StringWriter object. The buffer containing the string is initialized to the given size.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.StringWriter(
    int initialSize);
```

Parameters

initialSize

The length for the buffer that will store the array of characters. This value must be greater than zero.

See Also

Reference

StringWriter Class

Concepts

StringWriter Members

java.io Package

StringWriter Methods

Public Methods

Name	Description
close	Overridden. Closes the writer once the string has been processed.
equals	(inherited from Object)
flush	Overridden. Flushes the buffer representing the string to be written.
getBuffer	Retrieves the StringBuffer object used to store the string.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader. (inherited from Writer)
toString	Overridden. Displays a human readable representation of a StringWriter object. The contents of the buffer holding t he string is returned.
write	Overloaded. Appends a character or characters to the end of the string buffer.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringWriter Class

Concepts

java.io Package

StringWriter.close Method

Closes the writer once the string has been processed.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void close();
```

Example

The following example demonstrates how to close a StringWriter object.

```
// stringwriter_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: Hello World
*/
```

Remarks

This method does nothing for StringWriter objects.

See Also

Reference

StringWriter Class

Concepts

StringWriter Members

java.io Package

StringWriter.flush Method

Flushes the buffer representing the string to be written.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void flush();
```

Example

The following example demonstrates that the flush method for StringWriter objects has no effect on the underlying buffer.

```
// stringwriter_flush.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Now flush the buffer we just populated.
        writer.flush();

        // Print out the contents of the StringWriter buffer.
        System.out.println("The post-flush StringWriter buffer " +
            "contains: " + writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: Hello World
The post-flush StringWriter buffer contains: Hello World
*/
```

Remarks

This method does nothing for StringWriter objects.

See Also

Reference

StringWriter Class

Concepts

StringWriter.getBuffer Method

Retrieves the StringBuffer object used to store the string.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer getBuffer();
```

Return Value

The StringBuffer object used to store the string. This object is not a copy of the buffer that this writer is writing into. Any modifications made to the buffer will modify the buffer that this writer is using.

Example

The following example shows how to display the contents of the StringWriter buffer.

```
// stringwriter_getbuffer.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: Hello World
*/
```

See Also

Reference

StringWriter Class

Concepts

StringWriter Members

java.io Package

StringWriter.toString Method

Displays a human readable representation of a StringWriter object. The contents of the buffer holding the string is returned.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

A human readable representation of a StringWriter object. The contents of the buffer holding the string is returned.

See Also

Reference

StringWriter Class

Concepts

StringWriter Members

java.io Package

StringWriter.write Method

Appends a character or characters to the end of the string buffer.

Overload List

Name	Description
StringWriter.write (char[])	Places the given array of characters into the internal buffer holding the characters.
StringWriter.write (int)	Appends a character to the end of the string buffer.
StringWriter.write (String)	Appends a string of characters to the end of the string buffer.
StringWriter.write (char[], int, int)	Appends the given string, starting from the offset and for the length specified, to the end of t he string buffer.
StringWriter.write (String, int, int)	Appends a string, starting from the offset and for the length specified, to the end of the strin g buffer.

See Also

Reference

StringWriter Class

Concepts

StringWriter Members

java.io Package

StringWriter.write Method (Int32)

Appends a character to the end of the string buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int oneChar);
```

Parameters

oneChar

The character to append to the end of the string buffer.

Example

The following example shows how to write one character at a time to a StringWriter object.

```
// stringwriter_write.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: Hello World
*/
```

See Also

Reference

StringWriter Class

Concepts

StringWriter Members

java.io Package

StringWriter.write Method (String)

Appends a string of characters to the end of the string buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str);
```

Parameters

str

The string to append to the end of the string buffer.

Example

The following example shows how to write a string to a StringWriter object.

```
// stringwriter_write_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        String s = "Hello world";
        writer.write(s);

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: Hello World
*/
```

See Also

Reference

StringWriter Class

Concepts

StringWriter Members

java.io Package

StringWriter.write Method (Char[ ], Int32, Int32)

Appends the given string, starting from the offset and for the length specified, to the end of the string buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    char[] b,
    int off,
    int len);
```

Parameters

b

The array of characters to append to the string buffer.

off

An offset into the array of characters. This value represents the index of the first character in the array to be written to the string buffer. This value must be greater than zero.

len

The number of characters to be written to the string buffer starting from the offset. This value plus the offset must be less than the overall length of the array.

Example

The following example shows how to write an array of characters to an underlying StringWriter buffer.

```
// stringwriter_write_3.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        // Start with the 7th character and write to the end.
        writer.write(arr, 6, arr.length - 6);

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: World
*/
```

See Also

Reference
StringWriter Class
Concepts
StringWriter Members
java.io Package

StringWriter.write Method (String, Int32, Int32)

Appends a string, starting from the offset and for the length specified, to the end of the string buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str,
    int off,
    int len);
```

Parameters

str

The string to append to the end of the string buffer.

off

An offset into the string. This value represents the index of the first character in the string to be written to the string buffer. This value must be greater than zero.

len

The number of characters to be written to the string buffer starting from the offset. This value plus the offset must be less than the overall length of the string.

Example

The following example shows how to write a string to an underlying StringWriter buffer.

```
// stringwriter_write_4.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        // Start with the 7th character and write to the end.
        String s = "Hello World";
        writer.write(s, 6, s.length() - 6);

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: World
*/
```

See Also

Reference

StringWriter Class

Concepts
StringWriter Members
java.io Package

SyncFailedException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.SyncFailedException
    extends java.io.IOException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.SyncFailedException

See Also

Concepts

SyncFailedException Members

java.io Package

SyncFailedException Members

The following tables list the members exposed by the SyncFailedException type.

Public Constructors

Name	Description
SyncFailedException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SyncFailedException Class

Concepts

java.io Package

SyncFailedException Constructor

Overload List

Name	Description
SyncFailedException (String)	
SyncFailedException (SerializationInfo, StreamingContext)	
SyncFailedException (String, Exception)	

See Also

Reference

SyncFailedException Class

Concepts

SyncFailedException Members

java.io Package

SyncFailedException Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.SyncFailedException(
    java.lang.String s);
```

Parameters

s

See Also

Reference

SyncFailedException Class

Concepts

SyncFailedException Members

java.io Package

SyncFailedException Constructor (SerializationInfo, StreamingContext)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.SyncFailedException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

SyncFailedException Class

Concepts

SyncFailedException Members

java.io Package

SyncFailedException Constructor (String, Exception)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.SyncFailedException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also
Reference
SyncFailedException Class
Concepts
SyncFailedException Members
java.io Package

SyncFailedException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SyncFailedException Class

Concepts

java.io Package

SyncFailedException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

SyncFailedException Class

Concepts

java.io Package

UnsupportedEncodingException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.UnsupportedEncodingException
    extends java.io.IOException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.UnsupportedEncodingException

See Also

Concepts

UnsupportedEncodingException Members

java.io Package

UnsupportedEncodingException Members

The following tables list the members exposed by the UnsupportedEncodingException type.

Public Constructors

Name	Description
UnsupportedEncodingException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UnsupportedEncodingException Class

Concepts

java.io Package

UnsupportedEncodingException Constructor

Overload List

Name	Description
UnsupportedEncodingException ()	
UnsupportedEncodingException (String)	
UnsupportedEncodingException (SerializationInfo, StreamingContext)	
UnsupportedEncodingException (String, Exception)	

See Also

Reference

UnsupportedEncodingException Class

Concepts

UnsupportedEncodingException Members

java.io Package

UnsupportedEncodingException Constructor ()

Initializes a new instance of the UnsupportedEncodingException Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.UnsupportedEncodingException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

UnsupportedEncodingException Class

Concepts

UnsupportedEncodingException Members

java.io Package

UnsupportedEncodingException Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.UnsupportedEncodingException(
    java.lang.String s);
```

Parameters

s

See Also

Reference

UnsupportedEncodingException Class

Concepts

UnsupportedEncodingException Members

java.io Package

UnsupportedEncodingException Constructor (SerializationInfo, StreamingContext)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.UnsupportedEncodingException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

UnsupportedEncodingException Class

Concepts

UnsupportedEncodingException Members

java.io Package

UnsupportedEncodingException Constructor (String, Exception)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.UnsupportedEncodingException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also
Reference
UnsupportedEncodingException Class
Concepts
UnsupportedEncodingException Members
java.io Package

UnsupportedEncodingException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UnsupportedEncodingException Class

Concepts

java.io Package

UnsupportedEncodingException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

UnsupportedEncodingException Class

Concepts

java.io Package

UTFDataFormatException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.UTFDataFormatException
    extends java.io.IOException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.UTFDataFormatException

See Also

Concepts

UTFDataFormatException Members

java.io Package

UTFDataFormatException Members

The following tables list the members exposed by the UTFDataFormatException type.

Public Constructors

Name	Description
UTFDataFormatException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UTFDataFormatException Class

Concepts

java.io Package

UTFDataFormatException Constructor

Overload List

Name	Description
UTFDataFormatException ()	
UTFDataFormatException (String)	
UTFDataFormatException (SerializationInfo, StreamingContext)	
UTFDataFormatException (String, Exception)	

See Also

Reference

UTFDataFormatException Class

Concepts

UTFDataFormatException Members

java.io Package

UTFDataFormatException Constructor ()

Initializes a new instance of the UTFDataFormatException Class .

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.UTFDataFormatException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

UTFDataFormatException Class

Concepts

UTFDataFormatException Members

java.io Package

UTFDataFormatException Constructor (String)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.UTFDataFormatException(
    java.lang.String s);
```

Parameters

s

See Also

Reference

UTFDataFormatException Class

Concepts

UTFDataFormatException Members

java.io Package

UTFDataFormatException Constructor (SerializationInfo, StreamingContext)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.UTFDataFormatException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

UTFDataFormatException Class

Concepts

UTFDataFormatException Members

java.io Package

UTFDataFormatException Constructor (String, Exception)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.UTFDataFormatException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also
Reference
UTFDataFormatException Class
Concepts
UTFDataFormatException Members
java.io Package

UTFDataFormatException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UTFDataFormatException Class

Concepts

java.io Package

UTFDataFormatException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

UTFDataFormatException Class

Concepts

java.io Package

WriteAbortedException Class

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public class java.io.WriteAbortedException
    extends java.io.ObjectStreamException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.io.IOException

java.io.ObjectStreamException

java.io.WriteAbortedException

See Also

Concepts

WriteAbortedException Members

java.io Package

WriteAbortedException Members

The following tables list the members exposed by the WriteAbortedException type.

Public Constructors

Name	Description
WriteAbortedException	Overloaded.

Public Fields

Name	Description
detail	

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	Overridden.
GetObjectData	Overridden.
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

WriteAbortedException Class

Concepts

java.io Package

WriteAbortedException Fields

Public Fields

Name	Description
detail	

See Also

Reference

WriteAbortedException Class

Concepts

java.io Package

WriteAbortedException.detail Field

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Exception detail;
```

See Also

Reference

WriteAbortedException Class

Concepts

WriteAbortedException Members

java.io Package

WriteAbortedException Constructor

Overload List

Name	Description
WriteAbortedException (SerializationInfo, StreamingContext)	
WriteAbortedException (String, Exception)	
WriteAbortedException (String, Exception)	

See Also

Reference

WriteAbortedException Class

Concepts

WriteAbortedException Members

java.io Package

WriteAbortedException Constructor (SerializationInfo, StreamingContext)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.WriteAbortedException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

WriteAbortedException Class

Concepts

WriteAbortedException Members

java.io Package

WriteAbortedException Constructor (String, Exception)

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.io.WriteAbortedException(
    java.lang.String msg,
    System.Exception inner);
```

Parameters

msg

inner

See Also

Reference

WriteAbortedException Class

Concepts

WriteAbortedException Members

java.io Package

WriteAbortedException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	Overridden.
GetObjectData	Overridden.
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

WriteAbortedException Class

Concepts

java.io Package

WriteAbortedException.getMessage Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getMessage();
```

See Also

Reference

WriteAbortedException Class

Concepts

WriteAbortedException Members

java.io Package

WriteAbortedException.GetObjectData Method

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

WriteAbortedException Class

Concepts

WriteAbortedException Members

java.io Package

WriteAbortedException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

WriteAbortedException Class

Concepts

java.io Package

Writer Class

An abstract class representing the base for all character writers in the java.io package.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.io.Writer
    extends java.lang.Object
```

Example

The following example demonstrates the close, flush, getBuffer, and write methods of the StringWriter class, which is a specialization of the abstract Writer class.

```
// writer_overview.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        // Start with the 7th character and write to the end.
        String s = "Hello World";
        writer.write("Hello World", 6, s.length() - 6);

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.toString());

        // Now flush the buffer we just populated.
        writer.flush();

        // Print out the contents of the StringWriter buffer.
        System.out.println("The post-flush StringWriter buffer " +
            "contains: " + writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: World
The post-flush StringWriter buffer contains: World
*/
```

Inheritance Hierarchy

java.lang.Object

java.io.Writer

Derived Classes

See Also

Concepts

Writer Members

An abstract class representing the base for all character writers in the java.io package.

The following tables list the members exposed by the Writer type.

Public Constructors

Name	Description
Writer	Overloaded. Initializes a new instance of a Writer object.

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer.

Public Methods

Name	Description
close	An abstract method that must be implemented by subclasses. This method should take care of flushing any internal buffers and resources when called.
equals	(inherited from Object)
flush	Flushes the buffer representing the characters to be written.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader.
toString	Overridden. Displays a human readable representation of a Writer object.
write	Overloaded. Writes the next character or characters to the characters written.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Writer Class

Concepts

java.io Package

Writer Fields

Public Fields

Name	Description
lock	The object used to synchronize operations on the writer.

See Also

Reference

Writer Class

Concepts

java.io Package

Writer.lock Field

The object used to synchronize operations on the writer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Object lock;
```

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer Constructor

Initializes a new instance of a Writer object.

Overload List

Name	Description
Writer ()	Initializes a new instance of a Writer object.
Writer (Object)	Initializes a new instance of a Writer object. The given lock is used to synchronize operations on the writer.

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer Constructor ()

Initializes a new instance of a Writer object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.Writer();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer Constructor (Object)

Initializes a new instance of a Writer object. The given lock is used to synchronize operations on the writer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
protected java.io.Writer(
    java.lang.Object lock);
```

Parameters

lock

The object used to synchronize operations on the reader.

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer Methods

Public Methods

Name	Description
close	An abstract method that must be implemented by subclasses. This method should take care of flushing any internal buffers and resources when called.
equals	(inherited from Object)
flush	Flushes the buffer representing the characters to be written.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a shallow copy of this reader.
toString	Overridden. Displays a human readable representation of a Writer object.
write	Overloaded. Writes the next character or characters to the characters written.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Writer Class

Concepts

java.io Package

Writer.clone Method

Creates a shallow copy of this reader.

Package: java.io

Assembly: vjslib (in vjslib.dll)

Return Value

A shallow copy of this reader.

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer.close Method

An abstract method that must be implemented by subclasses. This method should take care of flushing any internal buffers and resources when called.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void close() throws java.io.IOException;
```

Example

The following example demonstrates how to close a StringWriter object, which is a specialization of the abstract Writer class.

```
// writer_close.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: Hello World
*/
```

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer.flush Method

Flushes the buffer representing the characters to be written.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void flush() throws java.io.IOException;
```

Example

The following example demonstrates that the flush method for StringWriter objects has no effect on the underlying buffer. The StringWriter class is a specialization of the abstract Writer class.

```
// writer_flush.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Now flush the buffer we just populated.
        writer.flush();

        // Print out the contents of the StringWriter buffer.
        System.out.println("The post-flush StringWriter buffer " +
            "contains: " + writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: Hello World
The post-flush StringWriter buffer contains: Hello World
*/
```

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer.toString Method

Displays a human readable representation of a Writer object.

Package: java.io

Assembly: vjslib (in vjslib.dll)

Return Value

A human readable representation of a Writer object.

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer.write Method

Writes the next character or characters to the characters written.

Overload List

Name	Description
Writer.write (char[])	Places the given array of characters into the internal buffer holding the characters.
Writer.write (int)	Places a character into the next position of the internal character buffer.
Writer.write (String)	Places the given string into the internal buffer holding characters.
Writer.write (char[], int, int)	Places the given array of characters, starting from the offset and for the length specified, into the internal buffer holding the characters.
Writer.write (String, int, int)	Places the given string, starting from the offset and for the length specified, into the internal buffer holding the characters.

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer.write Method (Char[ ])

Places the given array of characters into the internal buffer holding the characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    char[] buf) throws java.io.IOException;
```

Parameters

buf

The array of characters to be written to the internal buffer.

See Also
Reference
Writer Class
Concepts
Writer Members
java.io Package

Writer.write Method (Int32)

Places a character into the next position of the internal character buffer.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    int oneChar) throws java.io.IOException;
```

Parameters

oneChar

The character to be placed into the next position of the internal buffer.

Example

The following example shows how to write one character at a time to a StringWriter object, which is a specialization of the abstract Writer class.

```
// writer_write.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        for (int i = 0; i < arr.length; i++)
        {
            writer.write(arr[i]);
        }

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: Hello World
*/
```

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer.write Method (String)

Places the given string into the internal buffer holding characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str) throws java.io.IOException;
```

Parameters

str

The string to be written to the internal buffer.

Example

The following example shows how to write a string to a StringWriter object, which is a specialization of the abstract Writer class.

```
// writer_write_2.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        String s = "Hello world";
        writer.write(s);

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: Hello World
*/
```

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer.write Method (Char[ ], Int32, Int32)

Places the given array of characters, starting from the offset and for the length specified, into the internal buffer holding the characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public abstract void write(
    char[] buf,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

buf

The array of characters to be written to the internal buffer.

offset

An offset into the array of characters. This value represents the index of the first character in the array to be written to the internal buffer. This value must be greater than zero.

count

The number of characters to be written to the internal buffer starting from the offset. This value plus the offset must be less than the overall length of the array.

Example

The following example shows how to write an array of characters to an underlying StringWriter buffer, which is a specialization of the abstract Writer class.

```
// writer_write_3.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        char[] arr = new char[] {'H', 'e', 'l', 'l', 'o',
            ' ', 'W', 'o', 'r', 'l', 'd' };

        // Start with the 7th character and write to the end.
        writer.write(arr, 6, arr.length - 6);

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: World
*/
```

See Also

Reference

Writer Class

Concepts

Writer Members

java.io Package

Writer.write Method (String, Int32, Int32)

Places the given string, starting from the offset and for the length specified, into the internal buffer holding the characters.

Package: java.io

Assembly: vjslib (in vjslib.dll)

```
public void write(
    java.lang.String str,
    int offset,
    int count) throws java.io.IOException;
```

Parameters

str

The string to be written to the internal buffer.

offset

An offset into the string. This value represents the index of the first character in the string to be written to the internal buffer. This value must be greater than zero.

count

The number of characters to be written to the internal buffer starting from the offset. This value plus the offset must be less than the overall length of the array.

Example

The following example shows how to write a string to an underlying StringWriter buffer, which is a specialization of the abstract Writer class.

```
// writer_write_4.jsl

import java.io.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a StringWriter object.
        StringWriter writer = new StringWriter();

        // Write some data to the StringWriter object.
        // Start with the 7th character and write to the end.
        String s = "Hello World";
        writer.write(s, 6, s.length() - 6);

        // Print out the contents of the StringWriter buffer.
        System.out.println("The StringWriter buffer contains: " +
            writer.getBuffer().toString());

        // Close the StringWriter buffer.
        writer.close();
    }
}

/*
Output:
The StringWriter buffer contains: World
*/
```

See Also

Reference

Writer Class
Concepts
Writer Members
java.io Package

java.lang

Contains the essential language classes and interfaces necessary for J# programs.

Classes

Class	Description
AbstractMethodError	The error that is thrown when the execution engine attempts to call a method that is declared abstract.
ArithmeticException	The exception that is thrown when an arithmetic error occurs, such as a division by zero.
ArrayIndexOutOfBoundsException	The exception that is thrown when an attempt is made to read beyond the bounds of an array.
ArrayStoreException	The Exception that is thrown when there is an error storing an object as an array, or if an attempt is made to store an invalid object in an array.
Boolean	The Boolean class wraps the primitive value boolean in an object. The class also contains methods for conversion and processing Boolean objects.
Byte	Wraps the primitive type byte into an object. It also contains methods for conversion and processing Byte objects.
Character	Wraps the char primitive type into an object. It contains method members to manipulate characters and to change their case.
ClassCastException	The exception that is thrown when attempting to cast an object to a class that it cannot be cast to.
ClassCircularityError	The error that is thrown when the execution engine encounters a circular inheritance within any class hierarchy.
ClassFormatError	The error that is thrown when the execution engine attempts to load a class that is improperly formatted.
ClassNotFoundException	The exception that is thrown when attempting to load a class using reflection that cannot be found.
CloneNotSupportedException	The exception that is thrown when attempting to clone an object that does not support cloning.
Double	The Double class wraps the primitive type double into an object. The class contains method members to perform tests and to convert from double to other types and vice versa.
Error	Represents the base error for all errors in the J# Class Libraries. An error indicates a problem in the underlying execution engine.
Exception	Represents the base exception for all exceptions in the J# Class Libraries. An exception indicates that an application-level error has occurred.
ExceptionInInitializerError	The error that is thrown when the execution engine encounters an exception during initialization of a class from the static initializer.

Float	The Float class wraps the primitive type float. It contains several methods for conversion between String and Float and processing Float objects.
IllegalAccessError	The error that is thrown when the execution engine attempts to access a member that it does not have permission to access.
IllegalAccessException	The exception that is thrown when attempting to access a member during reflection that the caller does not have permission to access.
IllegalArgumentException	The exception that is thrown when an invalid argument is provided to a method call.
IllegalStateException	The exception that is thrown when the program has entered an invalid state.
IllegalThreadStateException	The exception that is thrown when a thread has entered an illegal state.
IncompatibleClassChangeError	The error that is thrown when an incompatible change is made to a class.
IndexOutOfBoundsException	The exception that is thrown when an attempt is made to read beyond the bounds of a collection.
InstantiationError	The error that is thrown when the execution engine encounters a problem instantiating a new object.
InstantiationException	The exception that is thrown when an error occurs instantiating (creating) an object.
Integer	Declares an Integer class type that wraps the primitive type int into an object. The class contains method members to perform tests and to convert from int to other types and vice versa.
InternalError	An error thrown to indicate that an internal error has occurred.
InterruptedException	The exception that is thrown when a thread is interrupted.
LinkageError	The error that is thrown when there is an error linking an application.
Long	Wraps the primitive type long into an object. It also contains methods for conversion and processing Long objects.
Math	Declares the Math class type, which contains method members to perform basic numeric operations such as exponentiation, logarithms, and trigonometric functions.
NegativeArraySizeException	The exception that is thrown when attempting to create an array with less than zero elements.
NoClassDefFoundError	An error occurs when a class fails to load because no definition of this class was found.
NoSuchFieldError	An error thrown when the application tries to access a field of an object and that field does not exist anymore or has changed.
NoSuchFieldException	An exception thrown to indicate that a class doesn't contain a specific field.
NoSuchMethodError	An error that occurs when an application references a method that does not exist.
NoSuchMethodException	An exception thrown when an application tries to access a method that doesn't exist in the class.

NullPointerException	An exception thrown when you try to use a null value where an object should be used.
Number	An abstract class extended by all classes that represent numeric primitive types such as Byte, Short, Integer, Long, Float, and Double. The class contains abstract methods that convert the object value to any of the other numeric types. All these methods are overridden by the numeric subclasses. It also contains methods to convert numeric types to strings and vise versa.
NumberFormatException	An exception thrown when an invalid number format is encountered.
OutOfMemoryError	An error that occurs when your application runs out of memory.
RuntimeException	An exception thrown to indicate that a runtime error has occurred.
SecurityException	An exception thrown when a security error occurs.
Short	Wraps the primitive type short into an object. The class also contains methods for conversion and processing Short objects.
StackOverflowError	An error that is issued when a stack overflow occurs.
StringBuffer	Implements a sequence of characters, which can be modified by using the appropriate method calls. The main difference between a StringBuffer object and String object is that the latter is immutable. Therefore, it is recommended that you use StringBuffer to manipulate and update the string, and then convert it to a String object.
StringIndexOutOfBoundsException	An exception that is thrown by a string method when an index is greater than the size of the string
System	Contains useful fields such as the standard input, standard output, and standard error streams. It also contains methods to access properties, load files and libraries, and copy arrays.
UnknownError	An error that is issued when an unknown error occurs.
UnsatisfiedLinkError	An error that is issued when the execution engine cannot find the appropriate definition for a declared method.
UnsupportedOperationException	An exception that is thrown when an application tries to perform an unsupported operation.
VerifyError	An error issued when the execution engine cannot verify a class code because it contains issues related to class format semantics.
VirtualMachineError	An error issued to indicate that there is a problem with the execution engine.

Interfaces

Interface	Description
Comparable	Provides a method to compare two objects to determine if one object is less than, equal to, or greater than another object.

Runnable	An interface that abstracts an object that executes code when it is active such as a thread. A class that implements Runnable can run without inheriting the Thread class.

AbstractMethodError Class

The error that is thrown when the execution engine attempts to call a method that is declared abstract.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.AbstractMethodError
    extends java.lang.IncompatibleClassChangeError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.IncompatibleClassChangeError

java.lang.AbstractMethodError

See Also

Concepts

AbstractMethodError Members

java.lang Package

AbstractMethodError Members

The error that is thrown when the execution engine attempts to call a method that is declared abstract.

The following tables list the members exposed by the AbstractMethodError type.

Public Constructors

Name	Description
AbstractMethodError	Overloaded. Initializes a new instance of an AbstractMethodError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractMethodError Class

Concepts

java.lang Package

AbstractMethodError Constructor

Initializes a new instance of an AbstractMethodError object.

Overload List

Name	Description
AbstractMethodError ()	Initializes a new instance of an AbstractMethodError object.
AbstractMethodError (String)	Initializes a new instance of an AbstractMethodError object with the given message.
AbstractMethodError (SerializationInfo, StreamingContext)	Initializes a new instance of an AbstractMethodError object during deserialization.
AbstractMethodError (String, Exception)	Initializes a new instance of an AbstractMethodError object with the given message and inner exception.

See Also

Reference

AbstractMethodError Class

Concepts

AbstractMethodError Members

java.lang Package

AbstractMethodError Constructor ()

Initializes a new instance of an AbstractMethodError object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.AbstractMethodError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

AbstractMethodError Class

Concepts

AbstractMethodError Members

java.lang Package

AbstractMethodError Constructor (String)

Initializes a new instance of an AbstractMethodError object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.AbstractMethodError(
    java.lang.String s);
```

Parameters

s

A message describing the error condition.

See Also
Reference
AbstractMethodError Class
Concepts
AbstractMethodError Members
java.lang Package

AbstractMethodError Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an AbstractMethodError object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.AbstractMethodError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

AbstractMethodError Class

Concepts

AbstractMethodError Members

java.lang Package

AbstractMethodError Constructor (String, Exception)

Initializes a new instance of an AbstractMethodError object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.AbstractMethodError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the error condition.

inner

An inner exception that led to the AbstractMethodError.

See Also

Reference

AbstractMethodError Class

Concepts

AbstractMethodError Members

java.lang Package

AbstractMethodError Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractMethodError Class

Concepts

java.lang Package

AbstractMethodError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

AbstractMethodError Class

Concepts

java.lang Package

ArithmeticException Class

The exception that is thrown when an arithmetic error occurs, such as a division by zero.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.ArithmeticException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.ArithmeticException

See Also

Concepts

ArithmeticException Members

java.lang Package

ArithmeticException Members

The exception that is thrown when an arithmetic error occurs, such as a division by zero.

The following tables list the members exposed by the ArithmeticException type.

Public Constructors

Name	Description
ArithmeticException	Overloaded. Initializes a new instance of an ArithmeticException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various arithmetic exceptions to a java.lang.ArithmeticException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ArithmeticException Class

Concepts

java.lang Package

ArithmeticException Constructor

Initializes a new instance of an ArithmeticException object.

Overload List

Name	Description
ArithmeticException ()	Initializes a new instance of an ArithmeticException object.
ArithmeticException (String)	Initializes a new instance of an ArithmeticException object with the given message.
ArithmeticException (SerializationInfo, StreamingContext)	Initializes a new instance of an ArithmeticException object during deserialization.
ArithmeticException (String, Exception)	Initializes a new instance of an ArithmeticException object with the given message and inner exception.

See Also

Reference

ArithmeticException Class

Concepts

ArithmeticException Members

java.lang Package

ArithmeticException Constructor ()

Initializes a new instance of an ArithmeticException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArithmeticException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ArithmeticException Class

Concepts

ArithmeticException Members

java.lang Package

ArithmeticException Constructor (String)

Initializes a new instance of an ArithmeticException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArithmeticException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
ArithmeticException Class
Concepts
ArithmeticException Members
java.lang Package

ArithmeticException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an ArithmeticException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.ArithmeticException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

ArithmeticException Class

Concepts

ArithmeticException Members

java.lang Package

ArithmeticException Constructor (String, Exception)

Initializes a new instance of an ArithmeticException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArithmeticException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the ArithmeticException.

See Also

Reference

ArithmeticException Class

Concepts

ArithmeticException Members

java.lang Package

ArithmeticException Methods

Public Methods

Name	Description
<exceptFilter>	Maps various arithmetic exceptions to a java.lang.ArithmeticException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ArithmeticException Class

Concepts

java.lang Package

ArithmeticException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ArithmeticException Class

Concepts

java.lang Package

ArrayIndexOutOfBoundsException Class

The exception that is thrown when an attempt is made to read beyond the bounds of an array.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.ArrayIndexOutOfBoundsException
    extends java.lang.IndexOutOfBoundsException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.IndexOutOfBoundsException

java.lang.ArrayIndexOutOfBoundsException

See Also

Concepts

ArrayIndexOutOfBoundsException Members

java.lang Package

ArrayIndexOutOfBoundsException Members

The exception that is thrown when an attempt is made to read beyond the bounds of an array.

The following tables list the members exposed by the ArrayIndexOutOfBoundsException type.

Public Constructors

Name	Description
ArrayIndexOutOfBoundsException	Overloaded. Initializes a new instance of an ArrayIndexOutOfBoundsException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various index out of bounds exceptions to a java.lang.ArrayIndexOutOfBoundsException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ArrayIndexOutOfBoundsException Class

Concepts

java.lang Package

ArrayIndexOutOfBoundsException Constructor

Initializes a new instance of an ArrayIndexOutOfBoundsException object.

Overload List

Name	Description
ArrayIndexOutOfBoundsException ()	Initializes a new instance of an ArrayIndexOutOfBoundsException object.
ArrayIndexOutOfBoundsException (int)	Initializes a new instance of an ArrayIndexOutOfBoundsException object with the value of the index that caused the exception.
ArrayIndexOutOfBoundsException (String)	Initializes a new instance of an ArrayIndexOutOfBoundsException with the given message.
ArrayIndexOutOfBoundsException (SerializationInfo, StreamingContext)	Initializes a new instance of an ArrayIndexOutOfBoundsException object during deserialization.
ArrayIndexOutOfBoundsException (String, Exception)	Initializes a new instance of an ArrayIndexOutOfBoundsException object with the given message and inner exception.

See Also

Reference

ArrayIndexOutOfBoundsException Class

Concepts

ArrayIndexOutOfBoundsException Members

java.lang Package

ArrayIndexOutOfBoundsException Constructor ()

Initializes a new instance of an ArrayIndexOutOfBoundsException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArrayIndexOutOfBoundsException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ArrayIndexOutOfBoundsException Class

Concepts

ArrayIndexOutOfBoundsException Members

java.lang Package

ArrayIndexOutOfBoundsException Constructor (Int32)

Initializes a new instance of an ArrayIndexOutOfBoundsException object with the value of the index that caused the exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArrayIndexOutOfBoundsException(
    int i);
```

Parameters

i

The value of the index that caused the exception.

See Also

Reference

ArrayIndexOutOfBoundsException Class

Concepts

ArrayIndexOutOfBoundsException Members

java.lang Package

ArrayIndexOutOfBoundsException Constructor (String)

Initializes a new instance of an ArrayIndexOutOfBoundsException with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArrayIndexOutOfBoundsException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also

Reference

ArrayIndexOutOfBoundsException Class

Concepts

ArrayIndexOutOfBoundsException Members

java.lang Package

ArrayIndexOutOfBoundsException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an ArrayIndexOutOfBoundsException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.ArrayIndexOutOfBoundsException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
ArrayIndexOutOfBoundsException Class
Concepts
ArrayIndexOutOfBoundsException Members
java.lang Package

ArrayIndexOutOfBoundsException Constructor (String, Exception)

Initializes a new instance of an ArrayIndexOutOfBoundsException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArrayIndexOutOfBoundsException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the ArrayIndexOutOfBoundsException.

See Also
Reference
ArrayIndexOutOfBoundsException Class
Concepts
ArrayIndexOutOfBoundsException Members
java.lang Package

ArrayIndexOutOfBoundsException Methods

Public Methods

Name	Description
<exceptFilter>	Maps various index out of bounds exceptions to a java.lang.ArrayIndexOutOfBoundsException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ArrayIndexOutOfBoundsException Class

Concepts

java.lang Package

ArrayIndexOutOfBoundsException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ArrayIndexOutOfBoundsException Class

Concepts

java.lang Package

ArrayStoreException Class

The Exception that is thrown when there is an error storing an object as an array, or if an attempt is made to store an invalid object in an array.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.ArrayStoreException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.ArrayStoreException

See Also

Concepts

ArrayStoreException Members

java.lang Package

ArrayStoreException Members

The Exception that is thrown when there is an error storing an object as an array, or if an attempt is made to store an invalid object in an array.

The following tables list the members exposed by the ArrayStoreException type.

Public Constructors

Name	Description
ArrayStoreException	Overloaded. Initializes a new instance of an ArrayStoreException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various array store exceptions to a java.lang.ArrayStoreException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ArrayStoreException Class

Concepts

java.lang Package

ArrayStoreException Constructor

Initializes a new instance of an ArrayStoreException object.

Overload List

Name	Description
ArrayStoreException ()	Initializes a new instance of an ArrayStoreException object.
ArrayStoreException (String)	Initializes a new instance of an ArrayStoreException with the given message.
ArrayStoreException (SerializationInfo, StreamingContext)	Initializes a new instance of an ArrayStoreException object during deserialization.
ArrayStoreException (String, Exception)	Initializes a new instance of an ArrayStoreException object with the given message and inner exception.

See Also

Reference

ArrayStoreException Class

Concepts

ArrayStoreException Members

java.lang Package

ArrayStoreException Constructor ()

Initializes a new instance of an ArrayStoreException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArrayStoreException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ArrayStoreException Class

Concepts

ArrayStoreException Members

java.lang Package

ArrayStoreException Constructor (String)

Initializes a new instance of an ArrayStoreException with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArrayStoreException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
ArrayStoreException Class
Concepts
ArrayStoreException Members
java.lang Package

ArrayStoreException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an ArrayStoreException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.ArrayStoreException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
ArrayStoreException Class
Concepts
ArrayStoreException Members
java.lang Package

ArrayStoreException Constructor (String, Exception)

Initializes a new instance of an ArrayStoreException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ArrayStoreException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the ArrayStoreException.

See Also

Reference

ArrayStoreException Class

Concepts

ArrayStoreException Members

java.lang Package

ArrayStoreException Methods

Public Methods

Name	Description
<exceptFilter>	Maps various array store exceptions to a java.lang.ArrayStoreException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ArrayStoreException Class

Concepts

java.lang Package

ArrayStoreException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ArrayStoreException Class

Concepts

java.lang Package

Boolean Class

The Boolean class wraps the primitive value boolean in an object. The class also contains methods for conversion and processing Boolean objects.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Boolean
    extends java.lang.Object
    implements java.io.Serializable, System.IConvertible
```

Inheritance Hierarchy

java.lang.Object

java.lang.Boolean

See Also

Concepts

Boolean Members

java.lang Package

Boolean Members

The Boolean class wraps the primitive value boolean in an object. The class also contains methods for conversion and processing Boolean objects.

The following tables list the members exposed by the Boolean type.

Public Constructors

Name	Description
Boolean	Overloaded.

Public Fields

Name	Description
FALSE	The Boolean object that corresponds to the primitive type false.
TRUE	The Boolean object that corresponds to the primitive type true.
TYPE	The Class object that represents the primitive type boolean.

Public Methods

Name	Description
booleanValue	Returns the boolean primitive value stored in a Boolean object.
equals	Overridden. Checks if a specified object is equal to the current object.
getBoolean	Returns true if and only if the value of the system property named by the argument is equal, ignoring case, to the string "true".
hashCode	Overridden. Returns the hash code of a Boolean object.
getClass	(inherited from Object)
clone	
ToBoolean	
ToByte	
ToChar	
ToDateTime	
ToDecimal	
ToDouble	
ToInt16	
ToInt32	
ToInt64	

ToSByte	
ToSingle	
toString	Overloaded. Overridden.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Returns the Boolean value of a string parameter.

See Also

Reference

Boolean Class

Concepts

java.lang Package

Boolean Fields

Public Fields

Name	Description
FALSE	The Boolean object that corresponds to the primitive type false.
TRUE	The Boolean object that corresponds to the primitive type true.
TYPE	The Class object that represents the primitive type boolean.

See Also

Reference

Boolean Class

Concepts

java.lang Package

Boolean.FALSE Field

The Boolean object that corresponds to the primitive type false.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Boolean FALSE;
```

Example

```
// b-FALSE.jsl
// Boolean.FALSE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print(Boolean.FALSE);
    }
}

/*
Output:
false
*/
```

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.TRUE Field

The Boolean object that corresponds to the primitive type true.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Boolean TRUE;
```

Example

```
// b-TRUE.jsl
// Boolean.TRUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print(Boolean.TRUE);
    }
}

/*
Output:
true
*/
```

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.TYPE Field

The Class object that represents the primitive type boolean.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Class TYPE;
```

Example

```
// b-TYPE.jsl
// Boolean.TYPE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print(Boolean.TYPE);
    }
}

/*
Output:
boolean
*/
```

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean Constructor

Overload List

Name	Description
Boolean (boolean)	Constructs a Boolean object by using a Boolean primitive type.
Boolean (String)	Constructs a Boolean object by using a string expression.

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean Constructor (Boolean)

Constructs a Boolean object by using a Boolean primitive type.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Boolean(
    boolean val);
```

Parameters

val

A primitive type expression.

Example

```
// b-ctor1.jsl
// Boolean.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Boolean b = new Boolean(true);
        System.out.print(b);
    }
}

/*
Output:
true
*/
```

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean Constructor (String)

Constructs a Boolean object by using a string expression.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Boolean(
    java.lang.String str);
```

Parameters

str

A string expression.

Example

```
// b-ctor2.jsl
// Boolean.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Boolean b = new Boolean("true");

        System.out.print(b);
    }
}
/*
Output:
true
*/
```

Remarks

Creates an instance of Boolean with its underlying value true if the string is "true" (case ignored); false otherwise.

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean Methods

Public Methods

Name	Description
booleanValue	Returns the boolean primitive value stored in a Boolean object.
equals	Overridden. Checks if a specified object is equal to the current object.
getBoolean	Returns true if and only if the value of the system property named by the argument is equal, ignoring case, to th e string "true".
hashCode	Overridden. Returns the hash code of a Boolean object.
getClass	(inherited from Object)
clone	
ToBoolean	
ToByte	
ToChar	
ToDateTime	
ToDecimal	
ToDouble	
ToInt16	
ToInt32	
ToInt64	
ToSByte	
ToSingle	
toString	Overloaded. Overridden.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Returns the Boolean value of a string parameter.

See Also

Reference
Boolean Class
Concepts
java.lang Package

Boolean.booleanValue Method

Returns the boolean primitive value stored in a Boolean object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean booleanValue();
```

Return Value

The boolean primitive value stored in the Boolean object.

Example

```
// b-bvalue1.jsl
// Boolean.booleanValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Boolean b = new Boolean("false");
        System.out.print(b.booleanValue());
    }
}

/*
Output:
false
*/
```

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.clone Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.equals Method

Checks if a specified object is equal to the current object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Example

```
// b-equals1.jsl
// Boolean.Equals example

public class MyClass
{
    public static void main(String[] args)
    {
        Boolean b1 = new Boolean(false);
        Boolean b2 = new Boolean("false");
        System.out.print(b1.Equals(b2));
    }
}

/*
Output:
true
*/
```

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.getBoolean Method

Returns true if and only if the value of the system property named by the argument is equal, ignoring case, to the string "true".

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean getBoolean(
    java.lang.String name);
```

Parameters

name

The system property name.

Return Value

true if and only if the value of the system property named by the argument is equal, ignoring case, to the string "true"; false otherwise.

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.hashCode Method

Returns the hash code of a Boolean object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code of a Boolean object.

Example

```
// b-hashCode1.jsl
// Boolean.GetHashCode example

public class MyClass
{
    public static void main(String[] args)
    {
        Boolean b = new Boolean("true");
        int x = b.hashCode();
        System.out.print("The Hash code is: " + x);
    }
}

/*
Output:
The Hash code is: 1231
*/
```

Remarks

If this Boolean object represents true, the integer 1231 is returned.

If this Boolean object represents false, the integer 1237 is returned.

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToBoolean Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean ToBoolean(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Byte ToByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToChar Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public char ToChar(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToDateTime Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.DateTime ToDateTime(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToDecimal Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Decimal ToDecimal(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToDouble Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double ToDouble(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short ToInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int ToInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long ToInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToSByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte ToSByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToSingle Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float ToSingle(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.toString Method

Overload List

Name	Description
Boolean.ToString ()	Returns a string equal to "true" if this Boolean object represents true, and returns string equal to "false" if this Boolean object represents false.
Boolean.ToString (IFormatProvider)	

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.toString Method ()

Returns a string equal to "true" if this Boolean object represents true, and returns string equal to "false" if this Boolean object represents false.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

A string "true" if this Boolean object represents true; a string "false" otherwise.

Example

```
// b-ToString1.jsl
// Boolean.ToString example

public class MyClass
{
    public static void main(String[] args)
    {
        Boolean b = new Boolean("true");
        System.out.print(b.ToString());
    }
}

/*
Output:
true
*/
```

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.toString Method (IFormatProvider)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String ToString(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToType Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object ToType(
    System.Type conversionType,
    System.IFormatProvider provider);
```

Parameters

conversionType

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToUInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt16 ToUInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToUInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt32 ToUInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.ToUInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt64 ToUInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Boolean.valueOf Method

Returns the Boolean value of a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Boolean valueOf(
    java.lang.String str);
```

Parameters

str

The string parameter.

Return Value

The Boolean value of the string parameter.

Example

```
// b-valueof1.jsl
// Boolean.valueOf example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = "true";
        System.out.print(Boolean.valueOf(s));
    }
}

/*
Output:
true
*/
```

Remarks

The result is true if and only if the argument is not null and is equal, ignoring case, to the string "true".

See Also

Reference

Boolean Class

Concepts

Boolean Members

java.lang Package

Byte Class

Wraps the primitive type byte into an object. It also contains methods for conversion and processing Byte objects.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Byte
    extends java.lang.Number
    implements System.IConvertible, java.lang.Comparable
```

Remarks

This class implements **System.IConvertible**.

Inheritance Hierarchy

java.lang.Object

  java.lang.Number

   java.lang.Byte

See Also

Concepts

Byte Members

java.lang Package

Byte Members

Wraps the primitive type byte into an object. It also contains methods for conversion and processing Byte objects.

The following tables list the members exposed by the Byte type.

Public Constructors

Name	Description
Byte	Overloaded.

Public Fields

Name	Description
MAX_VALUE	A constant that stores the maximum value a byte can have.
MIN_VALUE	A constant that stores the minimum value a byte can have.
TYPE	A readonly value that represents the Class instance of the primitive type byte.

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value stored in a Byte object.
compareTo	Overloaded.
decode	Decodes a string parameter to a Byte object.
doubleValue	Overridden. Returns the value of a Byte object converted to double.
equals	Overridden. Compares the current Byte object to another object.
floatValue	Overridden. Returns value of a Byte object converted to float.
hashCode	Overridden. Returns the hash code of a Byte object.
getClass	(inherited from Object)
intValue	Overridden. Returns the value of a Byte object as an int.
longValue	Overridden. Returns the value of a Byte object as a long number.
clone	(inherited from Number)
parseByte	Overloaded.
shortValue	Overridden. Returns the value of a Byte object as a short number.
ToBoolean	
ToByte	
ToChar	

ToDateTime	
ToDecimal	
ToDouble	
ToInt16	
ToInt32	
ToInt64	
ToSByte	
ToSingle	
toString	Returns the string that represents a byte value.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Overloaded.

See Also

Reference

Byte Class

Concepts

java.lang Package

Byte Fields

Public Fields

Name	Description
MAX_VALUE	A constant that stores the maximum value a byte can have.
MIN_VALUE	A constant that stores the minimum value a byte can have.
TYPE	A readonly value that represents the Class instance of the primitive type byte.

See Also

Reference

Byte Class

Concepts

java.lang Package

Byte.MAX_VALUE Field

A constant that stores the maximum value a byte can have.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte MAX_VALUE;
```

Example

```
// bt-MAX_VALUE.jsl
// Byte MAX_VALUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("MAX_VALUE = " + Byte.MAX_VALUE);
    }
}

/*
Output:
MAX_VALUE = 127
*/
```

Remarks

The value of MAX_VALUE is 127.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.MIN_VALUE Field

A constant that stores the minimum value a byte can have.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte MIN_VALUE;
```

Example

```
// bt-MIN_VALUE.jsl
// Byte MIN_VALUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("MIN_VALUE = " + Byte.MIN_VALUE);
    }
}

/*
Output:
MIN_VALUE = -128
*/
```

Remarks

The value of MIN_VALUE is -128.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.TYPE Field

A readonly value that represents the Class instance of the primitive type byte.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Class TYPE;
```

Example

```
// bt-TYPE.jsl
// Byte.TYPE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("TYPE = " + Byte.TYPE);
    }
}

/*
Output:
TYPE = byte
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte Constructor

Overload List

Name	Description
Byte (byte)	Constructs a new Byte object that represents a specified byte value.
Byte (String)	Constructs a new Byte object that represents the byte value specified by a string parameter.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte Constructor (SByte)

Constructs a new Byte object that represents a specified byte value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Byte(
    byte value);
```

Parameters

value

A byte expression.

Example

```
// bt-ctor1.jsl
// Byte.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte((byte)123);   // Cast to convert int to byte.
        System.out.print("The byte value is: " + b);
    }
}

/*
Output:
The byte value is: 123
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte Constructor (String)

Constructs a new Byte object that represents the byte value specified by a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Byte(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

A string expression.

Example

```
// bt-ctor2.jsl
// Byte.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        System.out.print("The byte value is: " + b);
    }
}

/*
Output:
The byte value is: 123
*/
```

Remarks

If the string could not be parsed to a byte value, it throws the exception java.lang.NumberFormatException.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte Methods

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value stored in a Byte object.
compareTo	Overloaded.
decode	Decodes a string parameter to a Byte object.
doubleValue	Overridden. Returns the value of a Byte object converted to double.
equals	Overridden. Compares the current Byte object to another object.
floatValue	Overridden. Returns value of a Byte object converted to float.
hashCode	Overridden. Returns the hash code of a Byte object.
getClass	(inherited from Object)
intValue	Overridden. Returns the value of a Byte object as an int.
longValue	Overridden. Returns the value of a Byte object as a long number.
clone	(inherited from Number)
parseByte	Overloaded.
shortValue	Overridden. Returns the value of a Byte object as a short number.
ToBoolean	
ToByte	
ToChar	
ToDateTime	
ToDecimal	
ToDouble	
ToInt16	
ToInt32	
ToInt64	
ToSByte	
ToSingle	

toString	Returns the string that represents a byte value.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Overloaded.

See Also

Reference

Byte Class

Concepts

java.lang Package

Byte.byteValue Method

Returns the byte value stored in a Byte object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte byteValue();
```

Return Value

The byte value stored in the Byte object.

Example

```
// bt-bvalue1.jsl
// Byte.byteValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        System.out.print("The byte value is: " + b.byteValue());
    }
}

/*
Output:
The byte value is: 123
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.compareTo Method

Overload List

Name	Description
Byte.compareTo (ubyte)	Compares the current Byte object to another Byte object.
Byte.compareTo (Object)	Compares the current object to another object.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.compareTo Method (Byte)

Compares the current Byte object to another Byte object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Byte aByte);
```

Parameters

aByte

The Byte object to compare to.

Return Value

0 if the two objects are identical. Negative value if aByte is greater than the current object. Positive value if aByte is less than the current object.

Example

```
// bt-comp1.jsl
// Byte.compareTo example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b1 = new Byte("121");
        Byte b2 = new Byte("123");
        System.out.print("The result is: " + b1.compareTo(b2));
    }
}

/*
Output:
The result is: -1
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.compareTo Method (Object)

Compares the current object to another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

If obj is a Byte object, it behaves exactly like compareTo, in which case it returns:

0 if the two objects are identical.

Negative value if obj is greater than the current object.

Positive value if obj is less than the current object.

If obj is of another type other than Byte, the program throws the exception java.lang.ClassCastException.

Example

```
// bt-comp2.jsl
// Byte.compareTo example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("121");
        Integer i = new Integer("123");
        System.out.print("The result is: " + b.compareTo(i));
    }
}

/*
Output:
java.lang.ClassCastException: Specified cast is not valid.
   at java.lang.Byte.compareTo(Object obj)
   at MyClass.main(String[] args)
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.decode Method

Decodes a string parameter to a Byte object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Byte decode(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be decoded.

Return Value

The Byte object that contains the byte value of str.

Example

```
// bt-decode1.jsl
// Byte.decode example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("0x27");   // Hexadecimal number
        Byte b = Byte.decode(s);
        System.out.print("The decoded byte is: " + b);
    }
}

/*
Output:
The decoded byte is: 39
*/
```

Remarks

The string parameter str might contain decimal, hexadecimal, or octal digits.

Hexadecimal digits are preceded by "0x", 0X", or "#".

Octal digits are preceded by zero (0).

If the string cannot be parsed to a byte value, the exception java.lang.NumberFormatException is thrown.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.doubleValue Method

Returns the value of a Byte object converted to double.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double doubleValue();
```

Return Value

The value of the Byte object converted to double.

Example

```
// bt-doubleValue1.jsl
// Byte.doubleValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        // Convert it to double and add it to another double number:
        double d = b.doubleValue() + 321.4;
        System.out.print("The new double number is: " + d);
    }
}

/*
Output:
The new double number is: 444.4
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.equals Method

Compares the current Byte object to another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

true if the two objects are identical; false otherwise.

Example

```
// bt-equals1.jsl
// Byte.Equals example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b1 = new Byte("123");
        Byte b2 = new Byte("123");
        System.out.print("The result is: " + b1.Equals(b2));
    }
}

/*
The result is: true
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.floatValue Method

Returns value of a Byte object converted to float.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float floatValue();
```

Return Value

The value of the Byte object converted to float.

Example

```
// bt-floatValue1.jsl
// Byte.floatValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        // Convert it to float and add it to another float number:
        float f = b.floatValue() + 321.4F;
        System.out.print("The new float number is: " + f);
    }
}

/*
Output:
The new float number is: 444.4
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.hashCode Method

Returns the hash code of a Byte object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code of the Byte object.

Example

```
// bt-getHash1.jsl
// Byte.GetHashCode example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        int h = b.GetHashCode();
        System.out.print("The hash code is: " + h);
    }
}

/*
The hash code is: 123
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.intValue Method

Returns the value of a Byte object as an int.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int intValue();
```

Return Value

The value of the Byte object as an int.

Example

```
// bt-intValue1.jsl
// Byte.intValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        // Convert it and add it to another int:
        int i = b.intValue() + 321;
        System.out.print("The new int is: " + i);
    }
}

/*
Output:
The new int is: 444
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.longValue Method

Returns the value of a Byte object as a long number.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long longValue();
```

Return Value

The value of the Byte object as a long number.

Example

```
// bt-longValue1.jsl
// Byte.longValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        // Convert it and add it to another long number:
        long l = b.longValue() + 321L;
        System.out.print("The new long number is: " + l);
    }
}

/*
Output:
The new long number is: 444
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.parseByte Method

Overload List

Name	Description
Byte.parseByte (String)	Returns the byte value represented by a string parameter.
Byte.parseByte (String, int)	Returns the signed byte value represented by a string parameter in the specified radix.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.parseByte Method (String)

Returns the byte value represented by a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static byte parseByte(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

Return Value

The byte value represented by the string parameter.

Example

```
// bt-parse1.jsl
// Byte.parseByte example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("123");
        byte b = Byte.parseByte(s);
        System.out.print("The byte is: " + b);
    }
}

/*
Output:
The byte is: 123
*/
```

Remarks

The string characters must all be decimal digits except the first character, which can be a minus sign (-).

The returned value is a signed byte in the decimal radix.

If the string could not be parsed to a signed decimal byte, the exception java.lang.NumberFormatException is thrown.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.parseByte Method (String, Int32)

Returns the signed byte value represented by a string parameter in the specified radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static byte parseByte(
    java.lang.String str,
    int radix) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

radix

The radix used in parsing str.

Return Value

The byte value represented by str in the specified radix.

Example

```
// bt-parse2.jsl
// Byte.parseByte example

public class MyClass
{
    public static void main(String[] args)
    {
        String s1 = new String("127");   // Decimal
        String s2 = new String("7F");    // Hexadecimal

        byte b1 = Byte.parseByte(s1,10);
        byte b2 = Byte.parseByte(s2,16);
        System.out.print("b1 = " + b1 + "\nb2 = " + b2);
    }
}

/*
Output:
b1 = 127
b2 = 127
*/
```

Remarks

The string characters must all be digits in the specified radix except the first character, which can be a minus sign (-).

The exception java.lang.NumberFormatException is generated if any of the following conditions occurred:

The string parameter is null.

The length of the string parameter is zero ("").

If the value of the radix is not in the range Character.MIN_RADIX to Character.MAX_RADIX.

The string does not represent a byte value.

If any of the characters in the string, except the first character, which can be a minus sign, is not a valid digit in the specified radix.

See Also

Reference
Byte Class
Concepts
Byte Members
java.lang Package

Byte.shortValue Method

Returns the value of a Byte object as a short number.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short shortValue();
```

Return Value

The value of the Byte object as a short number.

Example

```
// bt-shvalue1.jsl
// Byte.shortValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        short s = b.shortValue();
        System.out.print("The short value is: " + s);
    }
}

/*
Output:
The short value is: 123
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToBoolean Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean ToBoolean(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Byte ToByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToChar Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public char ToChar(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToDateTime Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.DateTime ToDateTime(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToDecimal Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Decimal ToDecimal(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToDouble Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double ToDouble(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short ToInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int ToInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long ToInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToSByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte ToSByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToSingle Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float ToSingle(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.toString Method

Returns the string that represents a byte value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toString(
    byte bval);
```

Parameters

bval

A byte expression.

Return Value

The string that represents the value of bval.

Example

```
// bt-toStr1.jsl
// Byte.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        byte b = 127;
        String s = Byte.toString(b);
        System.out.print("The string is: " + s);
    }
}

/*
The result is: 127
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.toString Method (J#)

Overload List

Name	Description
Byte.ToString ()	Returns the string that represents the value of a Byte object.
Byte.ToString (IFormatProvider)	Returns the string that represents the value of a Byte object using a format provider.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.toString Method (J#) ()

Returns the string that represents the value of a Byte object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The string that represents the value of a Byte object.

Example

```
// bt-toStr2.jsl
// Byte.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        // Converting to a string and catenating to another string:
        String s = "The string is: " + b.toString();
        System.out.print(s);
    }
}

/*
The string is: 123
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.toString Method (J#) (IFormatProvider)

Returns the string that represents the value of a Byte object using a format provider.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String ToString(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The string that represents the value of a Byte object using a format provider.

Example

```
// bt-toStr3.jsl
// Byte.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        Byte b = new Byte("123");
        // Converting to a string using a format provider:
        String s = b.toString(null);
        System.out.print("The string is: " + s);
    }
}

/*
The string is: 123
*/
```

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToType Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object ToType(
    System.Type conversionType,
    System.IFormatProvider provider);
```

Parameters

conversionType

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToUInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt16 ToUInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToUInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt32 ToUInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.ToUInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt64 ToUInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.valueOf Method

Overload List

Name	Description
Byte.valueOf (String)	Returns the Byte object represented by a string parameter.
Byte.valueOf (String, int)	Returns the Byte object represented by a string parameter in the specified radix.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.valueOf Method (String)

Returns the Byte object represented by a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Byte valueOf(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

Return Value

The Byte object represented by str.

Example

```
// bt-ValOf1.jsl
// Byte.valueOf example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("-128");
        Byte b = Byte.valueOf(s);
        System.out.print("The byte is: " + b);
    }
}

/*
Output:
The byte is: -128
*/
```

Remarks

The string characters must all be decimal digits except the first character, which can be a minus sign (-).

If the string could not be parsed to a signed decimal byte, the exception java.lang.NumberFormatException is thrown.

See Also

Reference

Byte Class

Concepts

Byte Members

java.lang Package

Byte.valueOf Method (String, Int32)

Returns the Byte object represented by a string parameter in the specified radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Byte valueOf(
    java.lang.String str,
    int radix) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

radix

The radix used in parsing str.

Return Value

The Byte object represented by str in the specified radix.

Example

```
// bt-ValOf2.jsl
// Byte.valueOf example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("-80");
        Byte b = Byte.valueOf(s,16);
        System.out.print("The byte is: " + b);
    }
}

/*
Output:
The byte is: -128
*/
```

Remarks

The string characters must all be digits in the specified radix except the first character, which can be a minus sign (-).

The exception java.lang.NumberFormatException is generated if any of the following conditions occurred:

The string parameter is null.

The length of the string parameter is zero ("").

If the value of the radix is not in the range Character.MIN_RADIX to Character.MAX_RADIX.

The string does not represent a byte value.

If any of the characters in the string, except the first character, which can be a minus sign, is not a valid digit in the specified radix.

See Also

Reference

Byte Class

Concepts

Byte Members

Character Class

Wraps the char primitive type into an object. It contains method members to manipulate characters and to change their case.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Character
    extends java.lang.Object
    implements java.io.Serializable, System.IConvertible, java.lang.Comparable
```

Remarks

This class implements the interface **System.IConvertible** (in the .NET Framework Class Library).

Inheritance Hierarchy

java.lang.Object

java.lang.Character

See Also

Concepts

Character Members

java.lang Package

Character Members

Wraps the char primitive type into an object. It contains method members to manipulate characters and to change their case.

The following tables list the members exposed by the Character type.

Public Constructors

Name	Description
Character	Constructs a new Character object that represents a specified char parameter.

Public Fields

Name	Description
COMBINING_SPACING_MARK	A Unicode category constant that represents a COMBINING_SPACING_MARK Unicode character.
CONNECTOR_PUNCTUATION	A Unicode category constant that represents a CONNECTOR_PUNCTUATION Unicode character.
CONTROL	A Unicode category constant that represents a CONTROL Unicode character.
CURRENCY_SYMBOL	A Unicode category constant that represents a CURRENCY_SYMBOL Unicode character.
DASH_PUNCTUATION	A Unicode category constant that represents a DASH_PUNCTUATION Unicode character.
DECIMAL_DIGIT_NUMBER	A constant returned by getType as the general category of a Unicode character.
ENCLOSING_MARK	A Unicode category constant that represents a ENCLOSING_MARK Unicode character.
END_PUNCTUATION	A Unicode category constant that represents a END_PUNCTUATION Unicode character.
FORMAT	A Unicode category constant that represents a FORMAT Unicode character.
LETTER_NUMBER	A Unicode category constant that represents a LETTER_NUMBER Unicode character.
LINE_SEPARATOR	A Unicode category constant that represents a LINE_SEPARATOR Unicode character.
LOWERCASE_LETTER	A Unicode category constant that represents a LOWERCASE_LETTER Unicode character.
MATH_SYMBOL	A Unicode category constant that represents a MATH_SYMBOL Unicode character.
MAX_RADIX	The maximum radix usable in converting characters. This constant defines the maximum radix that can be used to convert a character to or from another radix.
MAX_VALUE	The maximum possible value of a character.
MIN_RADIX	The minimum radix usable in converting characters. This constant defines the minimum radix that can be used to convert a character to or from another radix.
MIN_VALUE	The minimum possible value of a character.
MODIFIER_LETTER	A Unicode category constant that represents a MODIFIER_LETTER Unicode character.

MODIFIER_SYMBOL	A Unicode category constant that represents a MODIFIER_SYMBOL Unicode character.
NON_SPACING_MARK	A Unicode category constant that represents a NON_SPACING_MARK Unicode character.
OTHER_LETTER	A Unicode category constant that represents an OTHER_LETTER Unicode character.
OTHER_NUMBER	A Unicode category constant that represents an OTHER_NUMBER Unicode character.
OTHER_PUNCTUATION	A Unicode category constant that represents an OTHER_PUNCTUATION Unicode character.
OTHER_SYMBOL	A Unicode category constant that represents an OTHER_SYMBOL Unicode character.
PARAGRAPH_SEPARATOR	A Unicode category constant that represents a PARAGRAPH_SEPARATOR Unicode character.
PRIVATE_USE	A Unicode category constant that represents a PRIVATE_USE Unicode character.
SPACE_SEPARATOR	A Unicode category constant that represents a SPACE_SEPARATOR Unicode character.
START_PUNCTUATION	A Unicode category constant that represents a START_PUNCTUATION Unicode character.
SURROGATE	A Unicode category constant that represents a SURROGATE Unicode character.
TITLECASE_LETTER	A Unicode category constant that represents a TITLECASE_LETTER Unicode character.
TYPE	A java.lang.Class representing the Class object for the primitive char type.
UNASSIGNED	A Unicode category constant that represents a UNASSIGNED Unicode character.
UPPERCASE_LETTER	A Unicode category constant that represents an UPPERCASE_LETTER Unicode character.

Public Methods

Name	Description
charValue	Returns the char value of a Character object.
compareTo	Overloaded.
digit	Returns the numeric value of a specific char in a specified number system.
equals	Overridden. Checks if a Character object is equal to another object.
forDigit	Returns the char that represents a specified digit in a specific radix.
hashCode	Overridden. Returns the hash code of the current object.
getNumericValue	Returns the Unicode numeric value that represents a specified char.
getType	Returns the numeric value that represents the general category of a char.
getClass	(inherited from Object)
isDefined	Checks if a character defined in the Unicode character set.
isDigit	Checks if a specified char is a digit.

isIdentifierIgnorable	Checks if a specified character is ignored in an identifier.
isISOControl	Checks if a specified char is an ISO control character.
isJavaIdentifierPart	Checks if a specified char can be part of a Java identifier other than the first.
isJavaIdentifierStart	Checks if a specified char can be the first character in a Java identifier.
isJavaLetter	Checks if a specified char can be the first character in a Java identifier.
isJavaLetterOrDigit	Checks if a specified char is a Java letter or digit.
isLetter	Checks if a specified char is a letter.
isLetterOrDigit	Checks if a specified char is a letter or digit.
isLowerCase	Checks if a specified char is a lowercase letter.
isSpace	Checks if a specified char is a space.
isSpaceChar	Checks if a specified char is a Unicode space character.
isTitleCase	Checks if a specified char is a titlecase character.
isUnicodeIdentifierPart	Checks if a specified char can be part of a Unicode identifier other than the first.
isUnicodeIdentifierStart	Checks if a specified char can be the first character in a Unicode identifier.
isUpperCase	Checks if a specified char is an uppercase letter.
isWhitespace	Checks if a specified char is white space.
clone	Creates a copy of the object. Inherited from java.lang.Object.
ToBoolean	Returns a Boolean object representing a Character object.
ToByte	Returns a Byte object representing a Character object.
ToChar	Returns a Char object representing a Character object.
ToDateTime	Returns a DateTime object representing a Character object.
ToDecimal	Returns a Decimal object representing a Character object.
ToDouble	Returns a Double object representing a Character object.
ToInt16	Returns an Int16 object representing a Character object.
ToInt32	Returns an Int32 object representing a Character object.
ToInt64	Returns an Int64 object representing a Character object.
toLowerCase	Converts a char to the lowercase.

ToSByte	Returns a SByte object representing a Character object.
ToSingle	Returns a Single object representing a Character object.
toString	Overridden.
toTitleCase	Converts a char to its titlecase as specified in the Unicode attribute table.
ToType	Returns a Type representing a Character object.
ToUInt16	Returns an UInt16 object representing a Character object.
ToUInt32	Returns an UInt32 object representing a Character object.
ToUInt64	Returns an UInt64 object representing a Character object.
toUpperCase	Converts a char to the uppercase.

See Also

Reference

Character Class

Concepts

java.lang Package

Character Fields

Public Fields

Name	Description
COMBINING_SPACING_MARK	A Unicode category constant that represents a COMBINING_SPACING_MARK Unicode character.
CONNECTOR_PUNCTUATION	A Unicode category constant that represents a CONNECTOR_PUNCTUATION Unicode character.
CONTROL	A Unicode category constant that represents a CONTROL Unicode character.
CURRENCY_SYMBOL	A Unicode category constant that represents a CURRENCY_SYMBOL Unicode character.
DASH_PUNCTUATION	A Unicode category constant that represents a DASH_PUNCTUATION Unicode character.
DECIMAL_DIGIT_NUMBER	A constant returned by getType as the general category of a Unicode character.
ENCLOSING_MARK	A Unicode category constant that represents a ENCLOSING_MARK Unicode character.
END_PUNCTUATION	A Unicode category constant that represents a END_PUNCTUATION Unicode character.
FORMAT	A Unicode category constant that represents a FORMAT Unicode character.
LETTER_NUMBER	A Unicode category constant that represents a LETTER_NUMBER Unicode character.
LINE_SEPARATOR	A Unicode category constant that represents a LINE_SEPARATOR Unicode character.
LOWERCASE_LETTER	A Unicode category constant that represents a LOWERCASE_LETTER Unicode character.
MATH_SYMBOL	A Unicode category constant that represents a MATH_SYMBOL Unicode character.
MAX_RADIX	The maximum radix usable in converting characters. This constant defines the maximum radix that can be used to convert a character to or from another radix.
MAX_VALUE	The maximum possible value of a character.
MIN_RADIX	The minimum radix usable in converting characters. This constant defines the minimum radix that can be used to convert a character to or from another radix.
MIN_VALUE	The minimum possible value of a character.
MODIFIER_LETTER	A Unicode category constant that represents a MODIFIER_LETTER Unicode character.
MODIFIER_SYMBOL	A Unicode category constant that represents a MODIFIER_SYMBOL Unicode character.
NON_SPACING_MARK	A Unicode category constant that represents a NON_SPACING_MARK Unicode character.
OTHER_LETTER	A Unicode category constant that represents an OTHER_LETTER Unicode character.
OTHER_NUMBER	A Unicode category constant that represents an OTHER_NUMBER Unicode character.

OTHER_PUNCTUATION	A Unicode category constant that represents an OTHER_PUNCTUATION Unicode character.
OTHER_SYMBOL	A Unicode category constant that represents an OTHER_SYMBOL Unicode character.
PARAGRAPH_SEPARATOR	A Unicode category constant that represents a PARAGRAPH_SEPARATOR Unicode character.
PRIVATE_USE	A Unicode category constant that represents a PRIVATE_USE Unicode character.
SPACE_SEPARATOR	A Unicode category constant that represents a SPACE_SEPARATOR Unicode character.
START_PUNCTUATION	A Unicode category constant that represents a START_PUNCTUATION Unicode character.
SURROGATE	A Unicode category constant that represents a SURROGATE Unicode character.
TITLECASE_LETTER	A Unicode category constant that represents a TITLECASE_LETTER Unicode character.
TYPE	A Unicode category constant that represents a TYPE Unicode character.
UNASSIGNED	A Unicode category constant that represents a UNASSIGNED Unicode character.
UPPERCASE_LETTER	A Unicode category constant that represents an UPPERCASE_LETTER Unicode character.

See Also

Reference

Character Class

Concepts

java.lang Package

Character.COMBINING_SPACING_MARK Field

A Unicode category constant that represents a COMBINING_SPACING_MARK Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte COMBINING_SPACING_MARK;
```

Remarks

The following table shows the Unicode characters and their categories. Each Unicode character belong a specific category that can be returned by the method getType.

The following constants are not included in the table:

MIN_VALUE

MAX_VALUE

MIN_RADIX

MAX_RADIX

See the corresponding topic for a definition of a specific constant.

Constant	Category value	Unicode character	Displayed as	Remarks
SPACE_SEPARATOR	12	" "	\u0020	Space
LINE_SEPARATOR	13	<?>	\u2028	Line Separator
PARAGRAPH_SEPARATOR	14	<?>	\u2029	Paragraph separator
UPPERCASE_LETTER	1	A	\u0041	Latin capital letter A
LOWERCASE_LETTER	2	a	\u0061	Latin small letter a
TITLECASE_LETTER	3	Dz	\u01F2	Latin capital letter D with small letter z
MODIFIER_LETTER	4	h	\u02B0	Modifier letter small h
OTHER_LETTER	5	<?>	\u01BB	Latin letter two with stroke
DECIMAL_DIGIT_NUMBER	9	0	\u0030	Digit zero
LETTER_NUMBER	10	<?>	\u16EE	Runic Arlaug
OTHER_NUMBER	11	2	\u00B2	Superscript two
NON_SPACING_MARK	6	`	\u0300	Combining Grave accent
ENCLOSING_MARK	7	<?>	\u0488	Combining Cyrillic hundred thousand sign
COMBINING_SPACING_MARK	8	<?>	\u0903	Devnagari sign visarga

DASH_PUNCTUATION	20	-	\u002D	Hyphen-Minus
START_PUNCTUATION	21	(	\u0028	Left parenthesis
END_PUNCTUATION	22	)	\u0029	Right parenthesis
CONNECTOR_PUNCTUATION	23	<?>	\u005F	Low line (spacing underscore)
OTHER_PUNCTUATION	24	!	\u0021	Exclamation mark
MATH_SYMBOL	25	+	\u002B	Plus sign
CURRENCY_SYMBOL	26	$	\u0024	Dollar sign
MODIFIER_SYMBOL	27	^	\u005E	Circumflex accent
OTHER_SYMBOL	28	|	\u00A6	Broken bar
CONTROL	15	<control>	\u000A	Line feed
FORMAT	16	<?>	\u06DD	Arabic end of ayah
UNASSIGNED	0	<?>	\u0221	
PRIVATE_USE	18	<?>	\uE000	
SURROGATE	19	<?>	\uD800	

Notes:

<?> means that there is no suitable font to display this character

<control> means that it is a control character that has no type face.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.CONNECTOR_PUNCTUATION Field

A Unicode category constant that represents a CONNECTOR_PUNCTUATION Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte CONNECTOR_PUNCTUATION;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.CONTROL Field

A Unicode category constant that represents a CONTROL Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte CONTROL;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.CURRENCY_SYMBOL Field

A Unicode category constant that represents a CURRENCY_SYMBOL Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte CURRENCY_SYMBOL;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.DASH_PUNCTUATION Field

A Unicode category constant that represents a DASH_PUNCTUATION Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte DASH_PUNCTUATION;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.DECIMAL_DIGIT_NUMBER Field

A constant returned by getType as the general category of a Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte DECIMAL_DIGIT_NUMBER;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ENCLOSING_MARK Field

A Unicode category constant that represents a ENCLOSING_MARK Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte ENCLOSING_MARK;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.END_PUNCTUATION Field

A Unicode category constant that represents a END_PUNCTUATION Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte END_PUNCTUATION;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.FORMAT Field

A Unicode category constant that represents a FORMAT Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte FORMAT;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.LETTER_NUMBER Field

A Unicode category constant that represents a LETTER_NUMBER Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte LETTER_NUMBER;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.LINE_SEPARATOR Field

A Unicode category constant that represents a LINE_SEPARATOR Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte LINE_SEPARATOR;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.LOWERCASE_LETTER Field

A Unicode category constant that represents a LOWERCASE_LETTER Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte LOWERCASE_LETTER;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.MATH_SYMBOL Field

A Unicode category constant that represents a MATH_SYMBOL Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte MATH_SYMBOL;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.MAX_RADIX Field

The maximum radix usable in converting characters. This constant defines the maximum radix that can be used to convert a character to or from another radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final char MAX_RADIX;
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.MAX_VALUE Field

The maximum possible value of a character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final char MAX_VALUE;
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.MIN_RADIX Field

The minimum radix usable in converting characters. This constant defines the minimum radix that can be used to convert a character to or from another radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final char MIN_RADIX;
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.MIN_VALUE Field

The minimum possible value of a character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final char MIN_VALUE;
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.MODIFIER_LETTER Field

A Unicode category constant that represents a MODIFIER_LETTER Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte MODIFIER_LETTER;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.MODIFIER_SYMBOL Field

A Unicode category constant that represents a MODIFIER_SYMBOL Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte MODIFIER_SYMBOL;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.NON_SPACING_MARK Field

A Unicode category constant that represents a NON_SPACING_MARK Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte NON_SPACING_MARK;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.OTHER_LETTER Field

A Unicode category constant that represents an OTHER_LETTER Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte OTHER_LETTER;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.OTHER_NUMBER Field

A Unicode category constant that represents an OTHER_NUMBER Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte OTHER_NUMBER;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.OTHER_PUNCTUATION Field

A Unicode category constant that represents an OTHER_PUNCTUATION Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte OTHER_PUNCTUATION;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.OTHER_SYMBOL Field

A Unicode category constant that represents an OTHER_SYMBOL Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte OTHER_SYMBOL;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.PARAGRAPH_SEPARATOR Field

A Unicode category constant that represents a PARAGRAPH_SEPARATOR Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte PARAGRAPH_SEPARATOR;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.PRIVATE_USE Field

A Unicode category constant that represents a PRIVATE_USE Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte PRIVATE_USE;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.SPACE_SEPARATOR Field

A Unicode category constant that represents a SPACE_SEPARATOR Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte SPACE_SEPARATOR;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.START_PUNCTUATION Field

A Unicode category constant that represents a START_PUNCTUATION Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte START_PUNCTUATION;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.SURROGATE Field

A Unicode category constant that represents a SURROGATE Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte SURROGATE;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.TITLECASE_LETTER Field

A Unicode category constant that represents a TITLECASE_LETTER Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte TITLECASE_LETTER;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.TYPE Field

A java.lang.Class representing the Class object for the primitive char type.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Class TYPE;
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.UNASSIGNED Field

A Unicode category constant that represents a UNASSIGNED Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte UNASSIGNED;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.UPPERCASE_LETTER Field

A Unicode category constant that represents an UPPERCASE_LETTER Unicode character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final byte UPPERCASE_LETTER;
```

Example

See the table in COMBINING_SPACING_MARK.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character Constructor

Constructs a new Character object that represents a specified char parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Character(
    char ch);
```

Parameters

ch

A char parameter.

Example

This example constructs a new character object using the char value 'A'.

```
// c-ctor1.jsl
// Character ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Character c = new Character('A');
        System.out.print(c);
    }
}

/*
Output:
A
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character Methods

Public Methods

Name	Description
charValue	Returns the char value of a Character object.
compareTo	Overloaded.
digit	Returns the numeric value of a specific char in a specified number system.
equals	Overridden. Checks if a Character object is equal to another object.
forDigit	Returns the char that represents a specified digit in a specific radix.
hashCode	Overridden. Returns the hash code of the current object.
getNumericValue	Returns the Unicode numeric value that represents a specified char.
getType	Returns the numeric value that represents the general category of a char.
getClass	(inherited from Object)
isDefined	Checks if a character defined in the Unicode character set.
isDigit	Checks if a specified char is a digit.
isIdentifierIgnorable	Checks if a specified character is ignored in an identifier.
isISOControl	Checks if a specified char is an ISO control character.
isJavaIdentifierPart	Checks if a specified char can be part of a Java identifier other than the first.
isJavaIdentifierStart	Checks if a specified char can be the first character in a Java identifier.
isJavaLetter	Checks if a specified char can be the first character in a Java identifier.
isJavaLetterOrDigit	Checks if a specified char is a Java letter or digit.
isLetter	Checks if a specified char is a letter.
isLetterOrDigit	Checks if a specified char is a letter or digit.
isLowerCase	Checks if a specified char is a lowercase letter.
isSpace	Checks if a specified char is a space.
isSpaceChar	Checks if a specified char is a Unicode space character.
isTitleCase	Checks if a specified char is a titlecase character.
isUnicodeIdentifierPart	Checks if a specified char can be part of a Unicode identifier other than the first.

isUnicodeIdentifierStart	Checks if a specified char can be the first character in a Unicode identifier.
isUpperCase	Checks if a specified char is an uppercase letter.
isWhitespace	Checks if a specified char is white space.
clone	
ToBoolean	Returns a Boolean object representing a Character object.
ToByte	Returns a Byte object representing a Character object.
ToChar	Returns a Char object representing a Character object.
ToDateTime	Returns a DateTime object representing a Character object.
ToDecimal	Returns a Decimal object representing a Character object.
ToDouble	Returns a Double object representing a Character object.
ToInt16	Returns an Int16 object representing a Character object.
ToInt32	Returns an Int32 object representing a Character object.
ToInt64	Returns an Int64 object representing a Character object.
toLowerCase	Converts a char to the lowercase.
ToSByte	Returns a SByte object representing a Character object.
ToSingle	Returns a Single object representing a Character object.
toString	Overloaded. Overridden.
toTitleCase	Converts a char to its titlecase as specified in the Unicode attribute table.
ToType	Returns a Type representing a Character object.
ToUInt16	Returns an UInt16 object representing a Character object.
ToUInt32	Returns an UInt32 object representing a Character object.
ToUInt64	Returns an UInt64 object representing a Character object.
toUpperCase	Converts a char to the uppercase.

See Also

Reference

Character Class

Concepts

java.lang Package

Character.charValue Method

Returns the char value of a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public char charValue();
```

Return Value

The char value stored in the Character object.

Example

```
// c-charValue1.jsl
// Character.charValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Character obj = new Character('Z');
        char c = obj.charValue();
        System.out.print("The value of the object is " + c);
    }
}

/*
Output:
The value of the object is Z
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.clone Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.compareTo Method

Overload List

Name	Description
Character.compareTo (character)	Compares a specified Character object to the current Character object numerically.
Character.compareTo (Object)	Compares the current Character object to another object numerically.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.compareTo Method (Character)

Compares a specified Character object to the current Character object numerically.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Character aChar);
```

Parameters

aChar

The object to compare to.

Return Value

-1 if the parameter value is greater than the value of the current object. 0 if the two objects are identical. 1 if the parameter value is less than the value of the current object.

Example

```
// c-compareTo1.jsl
// Character.compareTo example

public class MyClass
{
    public static void main(String[] args)
    {
        Character obj1 = new Character('A');
        Character obj2 = new Character('Z');
        int result1 = obj1.compareTo(obj2);
        int result2 = obj1.compareTo(obj1);
        int result3 = obj2.compareTo(obj1);
        System.out.println("The comparison result is: " + result1);
        System.out.println("The comparison result is: " + result2);
        System.out.println("The comparison result is: " + result3);
    }
}

/*
Output:
The comparison result is: -1
The comparison result is: 0
The comparison result is: 1
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.compareTo Method (Object)

Compares the current Character object to another object numerically.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

If the two objects are of the type Character, the method behaves exactly like compareTo, and the returned values are: -1 if the parameter value is greater than the value of the current object. 0 if the two objects are identical. 1 if the parameter value is less than the value of the current object.

Example

```
See the example on compareTo.
```

Remarks

If the compared object is not a Character object the exception java.lang.ClassCastException is generated.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.digit Method

Returns the numeric value of a specific char in a specified number system.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int digit(
    char ch,
    int radix);
```

Parameters

ch

The char whose numeric value is returned.

radix

The radix of the specified number system.

Return Value

The numeric value of ch in the number system specified by radix.

Example

```
// c-digit1.jsl
// Character.digit example

public class MyClass
{
    public static void main(String[] args)
    {
        // The character:
        System.out.println('\u0042');
        // The numeric value:
        System.out.println(Character.digit('\u0042', 16));
    }
}

/*
Output:
B
11
*/
```

Remarks

This method returns the numeric value of ch if it is considered as a digit in the specified radix.

If the value of radix is not a valid radix, or ch is not a valid digit in the specified radix, then -1 is returned.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.equals Method

Checks if a Character object is equal to another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

true if the two objects are identical; false otherwise.

Example

```
// c-equals1.jsl
// Character.equals example

public class MyClass
{
    public static void main(String[] args)
    {
        boolean result1, result2;
        Character obj1 = new Character('A');
        Character obj2 = new Character('Z');
        result1 = obj1.equals(obj2);
        result2 = obj1.equals(obj1);

        System.out.println("The comparison result is: " + result1);
        System.out.println("The comparison result is: " + result2);
    }
}

/*
Output:
The comparison result is: false
The comparison result is: true
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.forDigit Method

Returns the char that represents a specified digit in a specific radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static char forDigit(
    int digit,
    int radix);
```

Parameters

digit

The int to be converted to char.

radix

The radix of the number system.

Return Value

The char that represents the specified digit in the specific radix.

Example

```
// c-foDigit1.jsl
// Character.forDigit example

public class MyClass
{
    public static void main(String[] args)
    {
        for (int i=0; i <= 15; i++)
        System.out.print(Character.forDigit(i, 16)+ " ");
    }
}

/*
Output:
0 1 2 3 4 5 6 7 8 9 a b c d e f
*/
```

Remarks

If the value of radix is not a valid radix, or the value of digit is not a valid digit in the specified radix, the null character is returned.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.hashCode Method

Returns the hash code of the current object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code of the current object.

Example

```
// c-HashCode1.jsl
// Character.GetHashCode example

public class MyClass
{
    public static void main(String[] args)
    {
        Character obj1 = new Character('A');
        Character obj2 = new Character('Z');
        int hc1 = obj1.GetHashCode();
        int hc2 = obj2.GetHashCode();

        System.out.println("The hash code of A is: " + hc1);
        System.out.println("The hash code of Z is: " + hc2);
    }
}

/*
Output:
The hash code of A is: 65
The hash code of Z is: 90
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.getNumericValue Method

Returns the Unicode numeric value that represents a specified char.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int getNumericValue(
    char ch);
```

Parameters

ch

The char whose Unicode value to be returned.

Return Value

The Unicode numeric value that represents the specified char.

Example

```
// c-GetNumVal1.jsl
// Character.getNumericVlaue example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The letter A = " +
                    Character.getNumericValue('\u0041'));
        System.out.println("The letter Z = " +
                    Character.getNumericValue('\u005A'));
        System.out.println("The fraction char u00BC returns: " +
                    Character.getNumericValue('\u00BC'));
        System.out.println(
        "The char u0F2A, which doesn't have a numeric value returns: "
         + Character.getNumericValue('\u0F2A'));
    }
}

/*
Output:
The letter A = 10
The letter Z = 35
The fraction char u00BC returns: -2
The char u0F2A, which doesn't have a numeric value returns: -1
*/
```

Remarks

This method returns the Unicode numeric value of the character as a nonnegative integer.

If the character does not have a numeric value, then -1 is returned.

If the character has a numeric value that cannot be represented as a nonnegative integer, for example, a fractional value, then -2 is returned.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.getType Method

Returns the numeric value that represents the general category of a char.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int getType(
    char ch);
```

Parameters

ch

The char whose general category to be returned.

Return Value

The numeric value that represents the general category of the char.

Example

```
// c-getType1.jsl
// Character.getType example

public class MyClass
{
    public static void main(String[] args)
    {
        int t1 = Character.getType('A');
        int t2 = Character.getType('a');

        System.out.println("The category of 'A' is: " + t1);
        System.out.println("The category of 'a' is: " + t2);
    }
}

/*
Output:
The category of 'A' is: 1
The category of 'a' is: 2
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isDefined Method

Checks if a character defined in the Unicode character set.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isDefined(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is defined in the Unicode character set; false otherwise.

Example

```
// c-isDefined1.jsl
// Character.isDefined example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(Character.isDefined('a'));
    }
}

/*
Output:
true
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isDigit Method

Checks if a specified char is a digit.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isDigit(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is a digit; false otherwise.

Example

```
// c-isDigit1.jsl
// Character.isDigit example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 ='A';
        char c2 = '3';
        boolean b1 = Character.isDigit(c1);
        boolean b2 = Character.isDigit(c2);
        if(b1 == true)
           System.out.println(c1 + " is a digit.");
        else
           System.out.println(c1 + " is not a digit.");
        if(b2 == true)
           System.out.println(c2 + " is a digit.");
        else
           System.out.println(c2 + " is not a digit.");
    }
}

/*
Output:
A is not a digit.
3 is a digit.
*/
```

Remarks

The character is a digit if its general category is DECIMAL_DIGIT_NUMBER.

The Unicode range from '\u0030' to '\u0039' contains the ISO-LATIN-1 digits from 0 to 9.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isIdentifierIgnorable Method

Checks if a specified character is ignored in an identifier.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isIdentifierIgnorable(
    char ch);
```

Parameters

ch

The char in to be checked.

Return Value

true if the char could be ignored in an identifier, false otherwise.

Example

```
// c-isIdIgn1.jsl
// isIdentifierIgnorable example

public class MyClass
{
    public static void main(String[] args)
    {
        // This char is not ignorable:
        System.out.println(Character.isIdentifierIgnorable('\u0009'));
        // This char is ignorable:
        System.out.println(Character.isIdentifierIgnorable('\u0007'));
    }
}

/*
Output:
false
true
*/
```

Remarks

The ignorable Unicode characters are control characters which are not whitespace:

from '\u0000') to '\u0008'

from '\u000E') to '\u001B'

from '\u007F' to '\u009F'

The Join controls:

from '\u200C' to '\u200F'

Bidirectional controls:

from '\u202A' to '\u202E'

Format controls:

from '\u206A' to '\u206F'

Zero-width no-break space

'\uFEFF'

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isISOControl Method

Checks if a specified char is an ISO control character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isISOControl(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is an ISO control character; false otherwise.

Example

```
// c-ISOcotl1.jsl
// Character.isISOControl example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(Character.isISOControl('\u008F'));
        System.out.println(Character.isISOControl('\u0000'));
        System.out.println(Character.isISOControl('\u010F'));
    }
}

/*
Output:
true
true
false
*/
```

Remarks

ISO control characters fall within the range \u0000 through \u001F or \u007F through \u009F.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isJavaIdentifierPart Method

Checks if a specified char can be part of a Java identifier other than the first.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isJavaIdentifierPart(
    char ch);
```

Parameters

ch

The char in to be checked.

Return Value

true if the char can be used as part of a Java identifier other than the first; false otherwise.

Example

```
// c-isIdPart1.jsl
// Character.isJavaIdentifierPart example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("It is "
        + Character.isJavaIdentifierPart('\u0041') +
        " that '\u0041' can be part of an identifier." );
        System.out.println("It is "
        + Character.isJavaIdentifierPart('\u0040') +
        " that '\u0040' can be part of an identifier." );
    }
}

/*
Output:
It is true that 'A' can be part of an identifier.
It is false that '@' can be part of an identifier.
*/
```

Remarks

The characters that are can be part of Java identifiers are:

Alphabetic letters.

The currency symbols, such as $.

The connecting punctuation characters, such as '_'.

Digits.

Combining-spacing-mark, non-spacing-mark and identifier-ignorable characters.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isJavaIdentifierStart Method

Checks if a specified char can be the first character in a Java identifier.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isJavaIdentifierStart(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char can be used as the first character in a Java identifier; false otherwise.

Example

```
// c-isIdstart1.jsl
// Character.isJavaIdentifierStart example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("It is "
        + Character.isJavaIdentifierStart('\u0041') +
        " that '\u0041' can be an identifier start." );
        System.out.println("It is "
        + Character.isJavaIdentifierStart('\u0040') +
        " that '\u0040' can be an identifier start." );
    }
}

/*
Output:
It is true that 'A' can be an identifier start.
It is false that '@' can be an identifier start.
*/
```

Remarks

The characters that are allowed to start identifiers are:

Letters.

The currency symbols, such as $.

The connecting punctuation characters, such as '_'.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isJavaLetter Method

Checks if a specified char can be the first character in a Java identifier.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isJavaLetter(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is a Java letter; false otherwise.

Example

```
// c-isJletter1.jsl
// Character.isJavaLetter example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 ='A';
        char c2 = '3';
        boolean b1 = Character.isJavaLetter(c1);
        boolean b2 = Character.isJavaLetter(c2);
        if(b1 == true)
           System.out.println(c1 + " is a Java letter.");
        else
           System.out.println(c1 + " is not a Java letter.");
        if(b2 == true)
           System.out.println(c2 + " is a Java letter.");
        else
           System.out.println(c2 + " is not a Java letter.");
    }
}

/*
Output:
A is a Java letter.
3 is not a Java letter.
*/
```

Remarks

This method is replaced by isJavaIdentifierStart.

The characters that are allowed to start identifiers are:

Letters.

The currency symbols, such as $.

The connecting punctuation characters, such as '_'.

See Also

Reference

Character Class

Concepts

Character Members
java.lang Package

Character.isJavaLetterOrDigit Method

Checks if a specified char is a Java letter or digit.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isJavaLetterOrDigit(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is a Java letter or digit; false otherwise.

Example

```
// c-isJlet-dig1.jsl
// Character.isJavaLetterOrDigitr example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 ='A';
        char c2 = '3';
        boolean b1 = Character.isJavaLetterOrDigit(c1);
        boolean b2 = Character.isJavaLetterOrDigit(c2);
        if(b1 == true)
           System.out.println(c1 + " is a Java letter/digit.");
        else
           System.out.println(c1 + " is not a Java letter/digit.");
        if(b2 == true)
           System.out.println(c2 + " is a Java letter/digit.");
        else
           System.out.println(c2 + " is not a Java letter/digit.");
    }
}

/*
Output:
A is a Java letter/digit.
3 is a Java letter/digit.
*/
```

Remarks

The characters that are can be part of Java identifiers are:

Alphabetic letters.

The currency symbols, such as $.

The connecting punctuation characters, such as '_'.

Digits.

Combining-spacing-mark, non-spacing-mark and identifier-ignorable characters.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isLetter Method

Checks if a specified char is a letter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isLetter(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is a letter; false otherwise.

Example

```
// c-isletter1.jsl
// Character.isLetter example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 ='A';
        char c2 = '3';
        boolean b1 = Character.isLetter(c1);
        boolean b2 = Character.isLetter(c2);
        if(b1 == true)
           System.out.println(c1 + " is a letter.");
        else
           System.out.println(c1 + " is not a letter.");
        if(b2 == true)
           System.out.println(c2 + " is a letter.");
        else
           System.out.println(c2 + " is not a letter.");
    }
}

/*
Output:
A is a letter.
3 is not a letter.
*/
```

Remarks

The character is a letter if its general category is one of the following: LOWERCASE_LETTER

UPPERCASE_LETTER

TITLECASE_LETTER

MODIFIER_LETTER

OTHER_LETTER

See Also

Reference

Character Class

Concepts

Character Members
java.lang Package

Character.isLetterOrDigit Method

Checks if a specified char is a letter or digit.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isLetterOrDigit(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is a letter or digit; false otherwise.

Example

```
// c-islet-dig1.jsl
// Character.isLetterOrDigitr example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 ='A';
        char c2 = '3';
        boolean b1 = Character.isLetterOrDigit(c1);
        boolean b2 = Character.isLetterOrDigit(c2);
        if(b1 == true)
           System.out.println(c1 + " is a letter/digit.");
        else
           System.out.println(c1 + " is not a letter/digit.");
        if(b2 == true)
           System.out.println(c2 + " is a letter/digit.");
        else
           System.out.println(c2 + " is not a letter/digit.");
    }
}

/*
Output:
A is a letter/digit.
3 is a letter/digit.
*/
```

Remarks

The charch is a letter or digit if one of the following methods returns true:

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isLowerCase Method

Checks if a specified char is a lowercase letter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isLowerCase(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is a lowercase letter; false otherwise.

Example

```
// c-islower1.jsl
// Character.isLowerCase example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 ='A';
        char c2 = 'a';
        boolean b1 = Character.isLowerCase(c1);
        boolean b2 = Character.isLowerCase(c2);
        if(b1 == true)
           System.out.println(c1 + " is lowercase.");
        else
           System.out.println(c1 + " is not lowercase.");
        if(b2 == true)
           System.out.println(c2 + " is lowercase.");
        else
           System.out.println(c2 + " is not lowercase.");
    }
}

/*
Output:
A is not lowercase.
a is lowercase.
*/
```

Remarks

The character is a lowercase letter if its general category is LOWERCASE_LETTER.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isSpace Method

Checks if a specified char is a space.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isSpace(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is a space; false otherwise.

Example

```
// c-isSpace1.jsl
// Character.isSpace example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 ='A';
        char c2 = ' ';
        boolean b1 = Character.isSpace(c1);
        boolean b2 = Character.isSpace(c2);
        if(b1 == true)
           System.out.println("'" + c1 + "'" + " is a space.");
        else
           System.out.println("'" + c1 + "'" + " is not a space.");
        if(b2 == true)
           System.out.println("'" + c2 + "'" + " is a space.");
        else
           System.out.println("'" + c2 + "'" + " is not a space.");
    }
}

/*
Output:
'A' is not a space.
' ' is a space.
*/
```

Remarks

This method is replaced by isWhitespace.

A character is space if it is one of the following:

Horizontal tab ('\u0009')

Line feed ('\u000A')

Form feed ('\u000C')

Carriage return ('\u000D')

A space ('\u0020')

See Also

Reference
Character Class
Concepts
Character Members
java.lang Package

Character.isSpaceChar Method

Checks if a specified char is a Unicode space character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isSpaceChar(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is a Unicode space character; false otherwise.

Example

```
// c-isSpChar1.jsl
// Character.isSpaceChar example

public class MyClass
{
    public static void main(String[] args)
    {
        // Testing an alphabetic character:
        System.out.println("It is "
        + Character.isSpaceChar('\u0042') +
        " that '\u0042' is a Unicode space char.");
        // Testing the Character.DASH_PUNCTUATION field:
        System.out.println("It is "
        + Character.isSpaceChar('\u0020') +
        " that '\u0020' is a Unicode space char.");
    }
}

/*
Output:
It is false that 'B' is a space char.
It is true that ' ' is a space char.
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isTitleCase Method

Checks if a specified char is a titlecase character.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isTitleCase(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is title case, false otherwise.

Example
Description

See the example on toTitleCase.

Remarks

The character is considered a titlecase character if the general category type of this character is TITLECASE_LETTER.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isUnicodeIdentifierPart Method

Checks if a specified char can be part of a Unicode identifier other than the first.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isUnicodeIdentifierPart(
    char ch);
```

Parameters

ch

The char in to be checked.

Return Value

true if the char can be used as part of a Unicode identifier other than the first; false otherwise.

Example

```
// c-isIdUPart1.jsl
// Character.isUnicodeIdentifierPart example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("It is "
        + Character.isUnicodeIdentifierPart('\u0041') +
        " that '\u0041' can be part of an identifier." );
        System.out.println("It is "
        + Character.isUnicodeIdentifierPart('\u0040') +
        " that '\u0040' can be part of an identifier." );
    }
}

/*
Output:
It is true that 'A' can be part of an identifier.
It is false that '@' can be part of an identifier.
*/
```

Remarks

The characters that are can be part of Unicode identifiers are:

Alphabetic letters.

The currency symbols, such as $.

The connecting punctuation characters, such as '_'.

Digits.

Combining-spacing-mark, non-spacing-mark and identifier-ignorable characters.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isUnicodeIdentifierStart Method

Checks if a specified char can be the first character in a Unicode identifier.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isUnicodeIdentifierStart(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the specified char can be the first character in a Unicode identifier; false otherwise.

Example

```
// c-isIdUStart1.jsl
// Character.isUnicodeIdentifierStart example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("It is "
        + Character.isUnicodeIdentifierStart('\u0041') +
        " that '\u0041' can be part of an identifier." );
        System.out.println("It is "
        + Character.isUnicodeIdentifierStart('\u0040') +
        " that '\u0040' can be part of an identifier." );
    }
}

/*
Output:
It is true that 'A' can be part of an identifier.
It is false that '@' can be part of an identifier.
*/
```

Remarks

The characters that are allowed to start Unicode identifiers are:

Alphabetic Letters.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isUpperCase Method

Checks if a specified char is an uppercase letter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isUpperCase(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is an uppercase letter; false otherwise.

Example

```
// c-isupper1.jsl
// Character.isUpperCase example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 ='A';
        char c2 = 'a';
        boolean b1 = Character.isUpperCase(c1);
        boolean b2 = Character.isUpperCase(c2);
        if(b1 == true)
           System.out.println(c1 + " is uppercase.");
        else
           System.out.println(c1 + " is not uppercase.");
        if(b2 == true)
           System.out.println(c2 + " is uppercase.");
        else
           System.out.println(c2 + " is not uppercase.");
    }
}

/*
Output:
A is uppercase.
a is not uppercase.
*/
```

Remarks

The character is an uppercase letter if its general category is UPPERCASE_LETTER.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.isWhitespace Method

Checks if a specified char is white space.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isWhitespace(
    char ch);
```

Parameters

ch

The char to be checked.

Return Value

true if the char is white space; false otherwise.

Example

```
// c-isWSpace1.jsl
// Character.isWhitespace example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 ='A';
        char c2 = ' ';
        boolean b1 = Character.isWhitespace(c1);
        boolean b2 = Character.isWhitespace(c2);
        if(b1 == true)
           System.out.println("'" + c1 + "'" + " is a white space.");
        else
           System.out.println("'" + c1 + "'" +
                              " is not a white space.");
        if(b2 == true)
           System.out.println("'" + c2 + "'" +
                              " is a white space.");
        else
           System.out.println("'" + c2 + "'" +
                              " is not a white space.");
    }
}

/*
Output:
'A' is not a white space.
' ' is a white space.
*/
```

Remarks

A character is a whitespace if its one of the following:

A Unicode category Zs ('\u3000'), Zl (\u2028),or Zp ('\u2029'), but not also a non-breaking space character such as \u00A0 or \u2007

Horizontal tab ('\u0009')

Line feed ('\u000A')

Vertical tab ('\u000B')

Form feed ('\u000C')

Carriage return ('\u000D')

File separator ('\u001C')

Group separator ('\u001D')

Record separator ('\u001E')

Unit separator ('\u001F')

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToBoolean Method

Returns a Boolean object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean ToBoolean(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Boolean object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToByte Method

Returns a Byte object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Byte ToByte(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Byte object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToChar Method

Returns a Char object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public char ToChar(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Char object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToDateTime Method

Returns a DateTime object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.DateTime ToDateTime(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A DateTime object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToDecimal Method

Returns a Decimal object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Decimal ToDecimal(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Decimal object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToDouble Method

Returns a Double object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double ToDouble(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Double object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToInt16 Method

Returns an Int16 object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short ToInt16(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An Int16 object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToInt32 Method

Returns an Int32 object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int ToInt32(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An Int32 object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToInt64 Method

Returns an Int64 object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long ToInt64(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An Int64 object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.toLowerCase Method

Converts a char to the lowercase.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static char toLowerCase(
    char ch);
```

Parameters

ch

The char to be converted.

Return Value

The lowercase char, if it was uppercase; the same char otherwise.

Example

```
// c-toLower1.jsl
// Character.toLowerCase example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 = Character.toLowerCase('A');
        char c2 = Character.toLowerCase('a');
        System.out.println(c1);
        System.out.println(c2);
    }
}

/*
Output:
a
a
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToSByte Method

Returns a SByte object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte ToSByte(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An SByte object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToSingle Method

Returns a Single object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float ToSingle(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Single object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.toString Method

Overload List

Name	Description
Character.ToString ()	Converts a Character object to a string.
Character.ToString (IFormatProvider)	Returns a String object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.toString Method ()

Converts a Character object to a string.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The char converted to a string.

Example

```
// c-toString1.jsl
// Character.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        Character c = new Character('A');

        // Convert to a String object:
        String s = c.toString();

        System.out.println("The first three letters are: " + s + "BC");
    }
}

/*
Output:
The first three letters are: ABC
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.toString Method (IFormatProvider)

Returns a String object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String ToString(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A String object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.toTitleCase Method

Converts a char to its titlecase as specified in the Unicode attribute table.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static char toTitleCase(
    char ch);
```

Parameters

ch

The char to be converted.

Return Value

The titlecase character of the parameter.

Example

```
// c-toTitlecase1.jsl
// Character.toTitleCase example

public class MyClass
{
    public static void main(String[] args)
    {
        // Converts to titlecase if exists, or to uppercase otherwise:
        System.out.println(Character.toTitleCase('i'));

        // The following statement true only if the character
        // is TitleCase:
        System.out.println(
            Character.isTitleCase(Character.toTitleCase('I')));
    }
}

/*
Output:
I
false
*/
```

Remarks

If the character has a titlecase equivalent specified in the Unicode attribute table, then that titlecase equivalent character is returned. Otherwise, if the character is a lowercase letter, the uppercase letter is returned, else the character itself is returned.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToType Method

Returns a Type representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object ToType(
    System.Type conversionType,
    System.IFormatProvider provider);
```

Parameters

conversionType

The Type to convert the Character object into.

provider

Provides information needed for formatting.

Return Value

An object of type conversionType representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToUInt16 Method

Returns an UInt16 object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt16 ToUInt16(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An UInt16 object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToUInt32 Method

Returns an UInt32 object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt32 ToUInt32(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An UInt32 object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.ToUInt64 Method

Returns an UInt64 object representing a Character object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt64 ToUInt64(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An UInt64 object representing a Character object.

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.toUpperCase Method

Converts a char to the uppercase.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static char toUpperCase(
    char ch);
```

Parameters

ch

The char to be converted.

Return Value

The uppercase char, if it was lowercase; the same char otherwise.

Example

```
// c-toUpper1.jsl
// Character.toUpperCase example

public class MyClass
{
    public static void main(String[] args)
    {
        char c1 = Character.toUpperCase('A');
        char c2 = Character.toUpperCase('a');
        System.out.println(c1);
        System.out.println(c2);
    }
}

/*
Output:
A
A
*/
```

See Also

Reference

Character Class

Concepts

Character Members

java.lang Package

Character.Subset Class

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.Character.Subset
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.lang.Character.Subset

java.lang.Character.UnicodeBlock

See Also

Concepts

Character.Subset Members

java.lang Package

Character.Subset Members

The following tables list the members exposed by the Character.Subset type.

Public Constructors

Name	Description
Character.Subset	

Public Methods

Name	Description
equals	Overridden.
hashCode	Overridden.
getClass	(inherited from Object)
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Character.Subset Class

Concepts

java.lang Package

Character.Subset Constructor

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Character.Subset(
    java.lang.String name);
```

Parameters

name

See Also

Reference

Character.Subset Class

Concepts

Character.Subset Members

java.lang Package

Character.Subset Methods

Public Methods

Name	Description
equals	Overridden.
hashCode	Overridden.
getClass	(inherited from Object)
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Character.Subset Class

Concepts

java.lang Package

Character.Subset.clone Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

See Also

Reference

Character.Subset Class

Concepts

Character.Subset Members

java.lang Package

Character.Subset.equals Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

Character.Subset Class

Concepts

Character.Subset Members

java.lang Package

Character.Subset.hashCode Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final int hashCode();
```

See Also

Reference

Character.Subset Class

Concepts

Character.Subset Members

java.lang Package

Character.Subset.toString Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String toString();
```

See Also

Reference

Character.Subset Class

Concepts

Character.Subset Members

java.lang Package

Character.UnicodeBlock Class

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Character.UnicodeBlock
    extends java.lang.Character$Subset
```

Inheritance Hierarchy

java.lang.Object

  java.lang.Character.Subset

   java.lang.Character.UnicodeBlock

See Also

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock Members

The following tables list the members exposed by the Character.UnicodeBlock type.

Public Fields

Name	Description
ALPHABETIC_PRESENTATION_FORMS	
ARABIC	
ARABIC_PRESENTATION_FORMS_A	
ARABIC_PRESENTATION_FORMS_B	
ARMENIAN	
ARROWS	
BASIC_LATIN	
BENGALI	
BLOCK_ELEMENTS	
BOPOMOFO	
BOX_DRAWING	
CJK_COMPATIBILITY	
CJK_COMPATIBILITY_FORMS	
CJK_COMPATIBILITY_IDEOGRAPHS	
CJK_SYMBOLS_AND_PUNCTUATION	
CJK_UNIFIED_IDEOGRAPHS	
COMBINING_DIACRITICAL_MARKS	
COMBINING_HALF_MARKS	
COMBINING_MARKS_FOR_SYMBOLS	
CONTROL_PICTURES	
CURRENCY_SYMBOLS	
CYRILLIC	
DEVANAGARI	

DINGBATS	
ENCLOSED_ALPHANUMERICS	
ENCLOSED_CJK_LETTERS_AND_MONTHS	
GENERAL_PUNCTUATION	
GEOMETRIC_SHAPES	
GEORGIAN	
GREEK	
GREEK_EXTENDED	
GUJARATI	
GURMUKHI	
HALFWIDTH_AND_FULLWIDTH_FORMS	
HANGUL_COMPATIBILITY_JAMO	
HANGUL_JAMO	
HANGUL_SYLLABLES	
HEBREW	
HIRAGANA	
IPA_EXTENSIONS	
KANBUN	
KANNADA	
KATAKANA	
LAO	
LATIN_1_SUPPLEMENT	
LATIN_EXTENDED_A	
LATIN_EXTENDED_ADDITIONAL	
LATIN_EXTENDED_B	
LETTERLIKE_SYMBOLS	
MALAYALAM	

MATHEMATICAL_OPERATORS	
MISCELLANEOUS_SYMBOLS	
MISCELLANEOUS_TECHNICAL	
NUMBER_FORMS	
OPTICAL_CHARACTER_RECOGNITION	
ORIYA	
PRIVATE_USE_AREA	
SMALL_FORM_VARIANTS	
SPACING_MODIFIER_LETTERS	
SPECIALS	
SUPERSCRIPTS_AND_SUBSCRIPTS	
SURROGATES_AREA	
TAMIL	
TELUGU	
THAI	
TIBETAN	

Public Methods

Name	Description
equals	(inherited from Character.Subset)
hashCode	(inherited from Character.Subset)
getClass	(inherited from Object)
clone	(inherited from Character.Subset)
of	
toString	(inherited from Character.Subset)

See Also

Reference

Character.UnicodeBlock Class

Concepts

java.lang Package

Character.UnicodeBlock Fields

Public Fields

Name	Description
ALPHABETIC_PRESENTATION_FORMS	
ARABIC	
ARABIC_PRESENTATION_FORMS_A	
ARABIC_PRESENTATION_FORMS_B	
ARMENIAN	
ARROWS	
BASIC_LATIN	
BENGALI	
BLOCK_ELEMENTS	
BOPOMOFO	
BOX_DRAWING	
CJK_COMPATIBILITY	
CJK_COMPATIBILITY_FORMS	
CJK_COMPATIBILITY_IDEOGRAPHS	
CJK_SYMBOLS_AND_PUNCTUATION	
CJK_UNIFIED_IDEOGRAPHS	
COMBINING_DIACRITICAL_MARKS	
COMBINING_HALF_MARKS	
COMBINING_MARKS_FOR_SYMBOLS	
CONTROL_PICTURES	
CURRENCY_SYMBOLS	
CYRILLIC	
DEVANAGARI	
DINGBATS	

ENCLOSED_ALPHANUMERICS	
ENCLOSED_CJK_LETTERS_AND_MONTHS	
GENERAL_PUNCTUATION	
GEOMETRIC_SHAPES	
GEORGIAN	
GREEK	
GREEK_EXTENDED	
GUJARATI	
GURMUKHI	
HALFWIDTH_AND_FULLWIDTH_FORMS	
HANGUL_COMPATIBILITY_JAMO	
HANGUL_JAMO	
HANGUL_SYLLABLES	
HEBREW	
HIRAGANA	
IPA_EXTENSIONS	
KANBUN	
KANNADA	
KATAKANA	
LAO	
LATIN_1_SUPPLEMENT	
LATIN_EXTENDED_A	
LATIN_EXTENDED_ADDITIONAL	
LATIN_EXTENDED_B	
LETTERLIKE_SYMBOLS	
MALAYALAM	
MATHEMATICAL_OPERATORS	

MISCELLANEOUS_SYMBOLS	
MISCELLANEOUS_TECHNICAL	
NUMBER_FORMS	
OPTICAL_CHARACTER_RECOGNITION	
ORIYA	
PRIVATE_USE_AREA	
SMALL_FORM_VARIANTS	
SPACING_MODIFIER_LETTERS	
SPECIALS	
SUPERSCRIPTS_AND_SUBSCRIPTS	
SURROGATES_AREA	
TAMIL	
TELUGU	
THAI	
TIBETAN	

See Also

Reference

Character.UnicodeBlock Class

Concepts

java.lang Package

Character.UnicodeBlock.ALPHABETIC_PRESENTATION_FORMS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock ALPHABETIC_PRESENTATION_FORMS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.ARABIC Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock ARABIC;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.ARABIC_PRESENTATION_FORMS_A Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock ARABIC_PRESENTATION_FORMS_A;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.ARABIC_PRESENTATION_FORMS_B Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock ARABIC_PRESENTATION_FORMS_B;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.ARMENIAN Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock ARMENIAN;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.ARROWS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock ARROWS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.BASIC_LATIN Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock BASIC_LATIN;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.BENGALI Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock BENGALI;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.BLOCK_ELEMENTS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock BLOCK_ELEMENTS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.BOPOMOFO Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock BOPOMOFO;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.BOX_DRAWING Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock BOX_DRAWING;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.CJK_COMPATIBILITY Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock CJK_COMPATIBILITY;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.CJK_COMPATIBILITY_FORMS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock CJK_COMPATIBILITY_FORMS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.CJK_COMPATIBILITY_IDEOGRAPHS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock CJK_COMPATIBILITY_IDEOGRAPHS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.CJK_SYMBOLS_AND_PUNCTUATION Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock CJK_SYMBOLS_AND_PUNCTUATION;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.CJK_UNIFIED_IDEOGRAPHS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock CJK_UNIFIED_IDEOGRAPHS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.COMBINING_DIACRITICAL_MARKS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock COMBINING_DIACRITICAL_MARKS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.COMBINING_HALF_MARKS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock COMBINING_HALF_MARKS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.COMBINING_MARKS_FOR_SYMBOLS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock COMBINING_MARKS_FOR_SYMBOLS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.CONTROL_PICTURES Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock CONTROL_PICTURES;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.CURRENCY_SYMBOLS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock CURRENCY_SYMBOLS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.CYRILLIC Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock CYRILLIC;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.DEVANAGARI Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock DEVANAGARI;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.DINGBATS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock DINGBATS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.ENCLOSED_ALPHANUMERICS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock ENCLOSED_ALPHANUMERICS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.ENCLOSED_CJK_LETTERS_AND_MONTHS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock ENCLOSED_CJK_LETTERS_AND_MONTHS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.GENERAL_PUNCTUATION Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock GENERAL_PUNCTUATION;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.GEOMETRIC_SHAPES Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock GEOMETRIC_SHAPES;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.GEORGIAN Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock GEORGIAN;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.GREEK Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock GREEK;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.GREEK_EXTENDED Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock GREEK_EXTENDED;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.GUJARATI Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock GUJARATI;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.GURMUKHI Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock GURMUKHI;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.HALFWIDTH_AND_FULLWIDTH_FORMS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock HALFWIDTH_AND_FULLWIDTH_FORMS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.HANGUL_COMPATIBILITY_JAMO Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock HANGUL_COMPATIBILITY_JAMO;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.HANGUL_JAMO Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock HANGUL_JAMO;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.HANGUL_SYLLABLES Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock HANGUL_SYLLABLES;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.HEBREW Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock HEBREW;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.HIRAGANA Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock HIRAGANA;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.IPA_EXTENSIONS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock IPA_EXTENSIONS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.KANBUN Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock KANBUN;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.KANNADA Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock KANNADA;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.KATAKANA Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock KATAKANA;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.LAO Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock LAO;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.LATIN_1_SUPPLEMENT Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock LATIN_1_SUPPLEMENT;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.LATIN_EXTENDED_A Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock LATIN_EXTENDED_A;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.LATIN_EXTENDED_ADDITIONAL Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock LATIN_EXTENDED_ADDITIONAL;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.LATIN_EXTENDED_B Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock LATIN_EXTENDED_B;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.LETTERLIKE_SYMBOLS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock LETTERLIKE_SYMBOLS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.MALAYALAM Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock MALAYALAM;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.MATHEMATICAL_OPERATORS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock MATHEMATICAL_OPERATORS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.MISCELLANEOUS_SYMBOLS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock MISCELLANEOUS_SYMBOLS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.MISCELLANEOUS_TECHNICAL Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock MISCELLANEOUS_TECHNICAL;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.NUMBER_FORMS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock NUMBER_FORMS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.OPTICAL_CHARACTER_RECOGNITION Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock OPTICAL_CHARACTER_RECOGNITION;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.ORIYA Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock ORIYA;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.PRIVATE_USE_AREA Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock PRIVATE_USE_AREA;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.SMALL_FORM_VARIANTS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock SMALL_FORM_VARIANTS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.SPACING_MODIFIER_LETTERS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock SPACING_MODIFIER_LETTERS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.SPECIALS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock SPECIALS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.SUPERSCRIPTS_AND_SUBSCRIPTS Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock SUPERSCRIPTS_AND_SUBSCRIPTS;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.SURROGATES_AREA Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock SURROGATES_AREA;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.TAMIL Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock TAMIL;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.TELUGU Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock TELUGU;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.THAI Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock THAI;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock.TIBETAN Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Character.UnicodeBlock TIBETAN;
```

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

Character.UnicodeBlock Methods

Public Methods

Name	Description
equals	(inherited from Character.Subset)
hashCode	(inherited from Character.Subset)
getClass	(inherited from Object)
clone	(inherited from Character.Subset)
of	
toString	(inherited from Character.Subset)

See Also

Reference

Character.UnicodeBlock Class

Concepts

java.lang Package

Character.UnicodeBlock.of Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Character$UnicodeBlock of(
    char c);
```

Parameters

c

See Also

Reference

Character.UnicodeBlock Class

Concepts

Character.UnicodeBlock Members

java.lang Package

ClassCastException Class

The exception that is thrown when attempting to cast an object to a class that it cannot be cast to.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.ClassCastException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.ClassCastException

See Also

Concepts

ClassCastException Members

java.lang Package

ClassCastException Members

The exception that is thrown when attempting to cast an object to a class that it cannot be cast to.

The following tables list the members exposed by the ClassCastException type.

Public Constructors

Name	Description
ClassCastException	Overloaded. Initializes a new instance of a ClassCastException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various invalid casting exceptions to a java.lang.ClassCastException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ClassCastException Class

Concepts

java.lang Package

ClassCastException Constructor

Initializes a new instance of a ClassCastException object.

Overload List

Name	Description
ClassCastException ()	Initializes a new instance of a ClassCastException object.
ClassCastException (String)	Initializes a new instance of a ClassCastException object with the given message.
ClassCastException (SerializationInfo, StreamingContext)	Initializes a new instance of a ClassCastException object during deserialization.
ClassCastException (String, Exception)	Initializes a new instance of a ClassCastException object with the given message and inner exception.

See Also

Reference

ClassCastException Class

Concepts

ClassCastException Members

java.lang Package

ClassCastException Constructor ()

Initializes a new instance of a ClassCastException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassCastException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ClassCastException Class

Concepts

ClassCastException Members

java.lang Package

ClassCastException Constructor (String)

Initializes a new instance of a ClassCastException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassCastException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
ClassCastException Class
Concepts
ClassCastException Members
java.lang Package

ClassCastException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a ClassCastException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.ClassCastException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

ClassCastException Class

Concepts

ClassCastException Members

java.lang Package

ClassCastException Constructor (String, Exception)

Initializes a new instance of a ClassCastException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassCastException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to a ClassCastException.

See Also

Reference

ClassCastException Class

Concepts

ClassCastException Members

java.lang Package

ClassCastException Methods

Public Methods

Name	Description
<exceptFilter>	Maps various invalid casting exceptions to a java.lang.ClassCastException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ClassCastException Class

Concepts

java.lang Package

ClassCastException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ClassCastException Class

Concepts

java.lang Package

ClassCircularityError Class

The error that is thrown when the execution engine encounters a circular inheritance within any class hierarchy.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.ClassCircularityError
    extends java.lang.LinkageError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.ClassCircularityError

See Also

Concepts

ClassCircularityError Members

java.lang Package

ClassCircularityError Members

The error that is thrown when the execution engine encounters a circular inheritance within any class hierarchy.

The following tables list the members exposed by the ClassCircularityError type.

Public Constructors

Name	Description
ClassCircularityError	Overloaded. Initializes a new instance of a ClassCircularityError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ClassCircularityError Class

Concepts

java.lang Package

ClassCircularityError Constructor

Initializes a new instance of a ClassCircularityError object.

Overload List

Name	Description
ClassCircularityError ()	Initializes a new instance of a ClassCircularityError object.
ClassCircularityError (String)	Initializes a new instance of a ClassCircularityError object with the gi ven message.
ClassCircularityError (SerializationInfo, StreamingContext)	Initializes a new instance of a ClassCircularityError object during de serialization.
ClassCircularityError (String, Exception)	Initializes a new instance of a ClassCircularityError object with the gi ven message and inner exception.

See Also

Reference

ClassCircularityError Class

Concepts

ClassCircularityError Members

java.lang Package

ClassCircularityError Constructor ()

Initializes a new instance of a ClassCircularityError object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassCircularityError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ClassCircularityError Class

Concepts

ClassCircularityError Members

java.lang Package

ClassCircularityError Constructor (String)

Initializes a new instance of a ClassCircularityError object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassCircularityError(
    java.lang.String s);
```

Parameters

s

A message describing the error condition.

See Also

Reference

ClassCircularityError Class

Concepts

ClassCircularityError Members

java.lang Package

ClassCircularityError Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a ClassCircularityError object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.ClassCircularityError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
ClassCircularityError Class
Concepts
ClassCircularityError Members
java.lang Package

ClassCircularityError Constructor (String, Exception)

Initializes a new instance of a ClassCircularityError object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassCircularityError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the error condition.

inner

An inner exception that led to the ClassCircularityError.

See Also

Reference

ClassCircularityError Class

Concepts

ClassCircularityError Members

java.lang Package

ClassCircularityError Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ClassCircularityError Class

Concepts

java.lang Package

ClassCircularityError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ClassCircularityError Class

Concepts

java.lang Package

ClassFormatError Class

The error that is thrown when the execution engine attempts to load a class that is improperly formatted.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.ClassFormatError
    extends java.lang.LinkageError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.ClassFormatError

See Also

Concepts

ClassFormatError Members

java.lang Package

ClassFormatError Members

The error that is thrown when the execution engine attempts to load a class that is improperly formatted.

The following tables list the members exposed by the ClassFormatError type.

Public Constructors

Name	Description
ClassFormatError	Overloaded. Initializes a new instance of a ClassFormatError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ClassFormatError Class

Concepts

java.lang Package

ClassFormatError Constructor

Initializes a new instance of a ClassFormatError object.

Overload List

Name	Description
ClassFormatError ()	Initializes a new instance of a ClassFormatError object.
ClassFormatError (String)	Initializes a new instance of a ClassFormatError object with the given message.
ClassFormatError (SerializationInfo, StreamingContext)	Initializes a new instance of a ClassFormatError object during deserialization.
ClassFormatError (String, Exception)	Initializes a new instance of a ClassFormatError object with the given message and inner exception.

See Also

Reference

ClassFormatError Class

Concepts

ClassFormatError Members

java.lang Package

ClassFormatError Constructor ()

Initializes a new instance of a ClassFormatError object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassFormatError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ClassFormatError Class

Concepts

ClassFormatError Members

java.lang Package

ClassFormatError Constructor (String)

Initializes a new instance of a ClassFormatError object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassFormatError(
    java.lang.String s);
```

Parameters

s

A message describing the error condition.

See Also

Reference

ClassFormatError Class

Concepts

ClassFormatError Members

java.lang Package

ClassFormatError Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a ClassFormatError object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.ClassFormatError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

ClassFormatError Class

Concepts

ClassFormatError Members

java.lang Package

ClassFormatError Constructor (String, Exception)

Initializes a new instance of a ClassFormatError object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassFormatError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the error condition.

inner

An inner exception that led to the ClassFormatError.

See Also
Reference
ClassFormatError Class
Concepts
ClassFormatError Members
java.lang Package

ClassFormatError Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ClassFormatError Class

Concepts

java.lang Package

ClassFormatError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ClassFormatError Class

Concepts

java.lang Package

ClassNotFoundException Class

The exception that is thrown when attempting to load a class using reflection that cannot be found.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.ClassNotFoundException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.ClassNotFoundException

See Also

Concepts

ClassNotFoundException Members

java.lang Package

ClassNotFoundException Members

The exception that is thrown when attempting to load a class using reflection that cannot be found.

The following tables list the members exposed by the ClassNotFoundException type.

Public Constructors

Name	Description
ClassNotFoundException	Overloaded. Initializes a new instance of a ClassNotFoundException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ClassNotFoundException Class

Concepts

java.lang Package

ClassNotFoundException Constructor

Initializes a new instance of a ClassNotFoundException object.

Overload List

Name	Description
ClassNotFoundException ()	Initializes a new instance of a ClassNotFoundException object.
ClassNotFoundException (String)	Initializes a new instance of a ClassNotFoundException object with the given message.
ClassNotFoundException (SerializationInfo, StreamingContext)	Initializes a new instance of a ClassNotFoundException object during deserialization.
ClassNotFoundException (String, Exception)	Initializes a new instance of a ClassNotFoundException object with the given message and inner exception.

See Also

Reference

ClassNotFoundException Class

Concepts

ClassNotFoundException Members

java.lang Package

ClassNotFoundException Constructor ()

Initializes a new instance of a ClassNotFoundException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassNotFoundException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ClassNotFoundException Class

Concepts

ClassNotFoundException Members

java.lang Package

ClassNotFoundException Constructor (String)

Initializes a new instance of a ClassNotFoundException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassNotFoundException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also

Reference

ClassNotFoundException Class

Concepts

ClassNotFoundException Members

java.lang Package

ClassNotFoundException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a ClassNotFoundException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.ClassNotFoundException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
ClassNotFoundException Class
Concepts
ClassNotFoundException Members
java.lang Package

ClassNotFoundException Constructor (String, Exception)

Initializes a new instance of a ClassNotFoundException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ClassNotFoundException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the ClassNotFoundException.

See Also
Reference
ClassNotFoundException Class
Concepts
ClassNotFoundException Members
java.lang Package

ClassNotFoundException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ClassNotFoundException Class

Concepts

java.lang Package

ClassNotFoundException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ClassNotFoundException Class

Concepts

java.lang Package

CloneNotSupportedException Class

The exception that is thrown when attempting to clone an object that does not support cloning.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.CloneNotSupportedException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.CloneNotSupportedException

See Also

Concepts

CloneNotSupportedException Members

java.lang Package

CloneNotSupportedException Members

The exception that is thrown when attempting to clone an object that does not support cloning.

The following tables list the members exposed by the CloneNotSupportedException type.

Public Constructors

Name	Description
CloneNotSupportedException	Overloaded. Initializes a new instance of a CloneNotSupportedException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

CloneNotSupportedException Class

Concepts

java.lang Package

CloneNotSupportedException Constructor

Initializes a new instance of a CloneNotSupportedException object.

Overload List

Name	Description
CloneNotSupportedException ()	Initializes a new instance of a CloneNotSupportedException object.
CloneNotSupportedException (String)	Initializes a new instance of a CloneNotSupportedException object with the given message.
CloneNotSupportedException (SerializationInfo, StreamingContext)	Initializes a new instance of a CloneNotSupportedException during deserialization.
CloneNotSupportedException (String, Exception)	Initializes a new instance of a CloneNotSupportedException with the given message and inner exception.

See Also

Reference

CloneNotSupportedException Class

Concepts

CloneNotSupportedException Members

java.lang Package

CloneNotSupportedException Constructor ()

Initializes a new instance of a CloneNotSupportedException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.CloneNotSupportedException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

CloneNotSupportedException Class

Concepts

CloneNotSupportedException Members

java.lang Package

CloneNotSupportedException Constructor (String)

Initializes a new instance of a CloneNotSupportedException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.CloneNotSupportedException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
CloneNotSupportedException Class
Concepts
CloneNotSupportedException Members
java.lang Package

CloneNotSupportedException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a CloneNotSupportedException during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.CloneNotSupportedException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
CloneNotSupportedException Class
Concepts
CloneNotSupportedException Members
java.lang Package

CloneNotSupportedException Constructor (String, Exception)

Initializes a new instance of a CloneNotSupportedException with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.CloneNotSupportedException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the CloneNotSupportedException.

See Also

Reference

CloneNotSupportedException Class

Concepts

CloneNotSupportedException Members

java.lang Package

CloneNotSupportedException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

CloneNotSupportedException Class

Concepts

java.lang Package

CloneNotSupportedException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

CloneNotSupportedException Class

Concepts

java.lang Package

Comparable Interface

Provides a method to compare two objects to determine if one object is less than, equal to, or greater than another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public interface java.lang.Comparable
```

Example

The following example shows how to implement and use the compareTo method of the Comparable interface. This example compares student's test scores.

```
// comparable.jsl

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create two new TestScore objects.
            TestScores bob = new TestScores("John Smith", 84);
            TestScores mary = new TestScores("Katie Jordan", 91);

            // Determine how bob compares to mary.
            int comparison = bob.compareTo(mary);

            // Print out the results of the comparison.
            System.out.print("John's test score was ");

            if (comparison == -1)
                System.out.print("less than ");
            else if (comparison == 0)
                System.out.print("equal to ");
            else
                System.out.print("greater than ");

            System.out.println("Katie's test score.");
        }
        catch (IllegalArgumentException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
John's test score was less than Katie's test score.
*/

// comparable_test_scores.jsl

public class TestScores implements Comparable
{
    public TestScores(String nameOfStudent,
                      int testScore)
    {
        studentName = nameOfStudent;
        score = testScore;
    }
```

```
    // Implements Comparable.compareTo
    // Compares the test scores of the two objects.
    public int compareTo(Object obj)
    {
        if (obj instanceof TestScores)
        {
            TestScores objTestScores = (TestScores)obj;
            if (score < objTestScores.score)
                return -1;
            else if (score > objTestScores.score)
                return 1;
            else
                return 0;
        }
        else
        {
            throw new IllegalArgumentException("obj must be an " +
                " instance of a TestScores object.");
        }
    }

    // The name of the student.
    private String studentName;

    // The score the student received on the test.
    private int score;
}
```

Remarks

Use the compareTo method of the Comparable interface to compare two objects of the same type. The rules for the return values for compareTo are summarized as follows:

Condition	Return Value
The object implementing compareTo is less than the parameter passed to compareTo.	-1
The object implementing compareTo is equal to the parameter passed to compareTo.	0
The object implementing compareTo is greater than the parameter passed to compareTo.	1

The method used to compare two objects is up to the developer of the class. Usually this is determined by comparing some or all of the fields of the class.

See Also

Concepts

Comparable Members

java.lang Package

Comparable Members

Provides a method to compare two objects to determine if one object is less than, equal to, or greater than another object.

The following tables list the members exposed by the Comparable type.

Public Methods

Name	Description
compareTo	Provides a means to compare two objects to determine if one object is less than, equal to, or greater than another object.

See Also

Reference

Comparable Interface

Concepts

java.lang Package

Comparable Methods

Public Methods

Name	Description
compareTo	Provides a means to compare two objects to determine if one object is less than, equal to, or greater than another object.

See Also

Reference

Comparable Interface

Concepts

java.lang Package

Comparable.compareTo Method

Provides a means to compare two objects to determine if one object is less than, equal to, or greater than another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract int compareTo(
    java.lang.Object obj);
```

Parameters

obj

The object to compare against.

Return Value

-1 if the object calling compareTo is less than obj, 0 if the two objects are equal, or 1 if the object calling compareTo is greater than obj.

Example

The following example shows how to implement and use the compareTo method of the Comparable interface. This example compares student's test scores.

```
// comparable.jsl

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create two new TestScore objects.
            TestScores bob = new TestScores("John Smith", 84);
            TestScores mary = new TestScores("Katie Jordan", 91);

            // Determine how bob compares to mary.
            int comparison = bob.compareTo(mary);

            // Print out the results of the comparison.
            System.out.print("John's test score was ");

            if (comparison == -1)
                System.out.print("less than ");
            else if (comparison == 0)
                System.out.print("equal to ");
            else
                System.out.print("greater than ");

            System.out.println("Katie's test score.");
        }
        catch (IllegalArgumentException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
John's test score was less than Katie's test score.
*/
```

```
// comparable_test_scores.jsl

public class TestScores implements Comparable
{
    public TestScores(String nameOfStudent,
                      int testScore)
    {
        studentName = nameOfStudent;
        score = testScore;
    }

    // Implements Comparable.compareTo
    // Compares the test scores of the two objects.
    public int compareTo(Object obj)
    {
        if (obj instanceof TestScores)
        {
            TestScores objTestScores = (TestScores)obj;
            if (score < objTestScores.score)
                return -1;
            else if (score > objTestScores.score)
                return 1;
            else
                return 0;
        }
        else
        {
            throw new IllegalArgumentException("obj must be an " +
                " instance of a TestScores object.");
        }
    }

    // The name of the student.
    private String studentName;

    // The score the student received on the test.
    private int score;
}
```

Remarks

Use the compareTo method of the Comparable interface to compare two objects of the same type. The rules for the return values for compareTo are summarized as follows:

Condition	Return Value
The object implementing compareTo is less than the parameter passed to compareTo.	-1
The object implementing compareTo is equal to the parameter passed to compareTo.	0
The object implementing compareTo is greater than the parameter passed to compareTo.	1

The method used to compare two objects is up to the developer of the class. Usually this is determined by comparing some or all of the fields of the class.

See Also

Reference

Comparable Interface

Concepts

Comparable Members

java.lang Package

Double Class

The Double class wraps the primitive type double into an object. The class contains method members to perform tests and to convert from double to other types and vice versa.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Double
    extends java.lang.Number
    implements System.IConvertible, java.lang.Comparable
```

Inheritance Hierarchy

java.lang.Object

  java.lang.Number

   java.lang.Double

See Also

Concepts

Double Members

java.lang Package

Double Members

The Double class wraps the primitive type double into an object. The class contains method members to perform tests and to convert from double to other types and vice versa.

The following tables list the members exposed by the Double type.

Public Constructors

Name	Description
Double	Overloaded. Constructs a Double object.

Public Fields

Name	Description
MAX_VALUE	
MIN_VALUE	
NaN	
NEGATIVE_INFINITY	
POSITIVE_INFINITY	
TYPE	

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value of a Double object.
compareTo	Overloaded.
doubleToLongBits	Returns the 64-bit representation of a double number as per IEEE 754 floating point standard.
doubleValue	Overridden. Returns the double value of a Double object.
equals	Overridden. Compares the current Double object to another object.
floatValue	Overridden. Returns the float value of a Double object.
hashCode	Overridden. Retrieves the hash code of the current Double object.
getClass	(inherited from Object)
intValue	Overridden. Returns the int value of a Doublee object.
isInfinite	Overloaded. Checks if the value of a Double object is infinite.
isNaN	Overloaded. Checks if the value of a Double object is "Not a Number."
longBitsToDouble	Returns double number that corresponds to the 64-bit representation of a double number.

longValue	Overridden. Returns the long value of a Double object.
clone	(inherited from Number)
parseDouble	Converts a string to a double value.
shortValue	Overridden. Returns the short value of a Double object.
ToBoolean	Converts a Double object value to a Boolean value using a specific format.
ToByte	Converts a Double object value to a byte using a specific format.
ToChar	Converts a Double object value to a char using a specific format.
ToDateTime	Converts a Double object value to a date-time value using a specific format.
ToDecimal	Converts a Double object value to a decimal value using a specific format.
ToDouble	
ToInt16	
ToInt32	
ToInt64	
ToSByte	
ToSingle	
toString	
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	

See Also

Reference

Double Class

Concepts

java.lang Package

Double Fields

Public Fields

Name	Description
MAX_VALUE	
MIN_VALUE	
NaN	
NEGATIVE_INFINITY	
POSITIVE_INFINITY	
TYPE	

See Also

Reference

Double Class

Concepts

java.lang Package

Double.MAX_VALUE Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final double MAX_VALUE;
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.MIN_VALUE Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final double MIN_VALUE;
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.NaN Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final double NaN;
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.NEGATIVE_INFINITY Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final double NEGATIVE_INFINITY;
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.POSITIVE_INFINITY Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final double POSITIVE_INFINITY;
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.TYPE Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Class TYPE;
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double Constructor

Constructs a Double object.

Overload List

Name	Description
Double (double)	Constructs a Double object an initializes it with a double parameter.
Double (String)	Constructs a Double object an initializes it with a string parameter.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double Constructor (Double)

Constructs a Double object an initializes it with a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Double(
    double d);
```

Parameters

d

A double expression.

Example

The following example creates an instance of the Double class and initializes it with the value 23.

```
// Double1.jsl
// Double constructor example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d =  new Double(23);

        System.out.print("The double number is: "+ d);
    }
}
/*
Output:
The double number is: 23.0
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double Constructor (String)

Constructs a Double object an initializes it with a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Double(
    java.lang.String s) throws java.lang.NumberFormatException;
```

Parameters

s

A string parameter.

Example

This example creates an instance of the Double class and initializes it with the string "22.0".

```
// Double2.jsl
// Double constructor example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d =  new Double("22.0");

        System.out.print("The double number is: "+ d);
    }
}

/*
Output:
The double number is: 22.0
*/
```

Remarks

The string parameter should contain a double expression.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double Methods

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value of a Double object.
compareTo	Overloaded.
doubleToLongBits	Returns the 64-bit representation of a double number as per IEEE 754 floating point standard.
doubleValue	Overridden. Returns the double value of a Double object.
equals	Overridden. Compares the current Double object to another object.
floatValue	Overridden. Returns the float value of a Double object.
hashCode	Overridden. Retrieves the hash code of the current Double object.
getClass	(inherited from Object)
intValue	Overridden. Returns the int value of a Doublee object.
isInfinite	Overloaded. Checks if the value of a Double object is infinite.
isNaN	Overloaded. Checks if the value of a Double object is "Not a Number."
longBitsToDouble	Returns double number that corresponds to the 64-bit representation of a double number.
longValue	Overridden. Returns the long value of a Double object.
clone	(inherited from Number)
parseDouble	Converts a string to a double value.
shortValue	Overridden. Returns the short value of a Double object.
ToBoolean	Converts a Double object value to a Boolean value using a specific format.
ToByte	Converts a Double object value to a byte using a specific format.
ToChar	Converts a Double object value to a char using a specific format.
ToDateTime	Converts a Double object value to a date-time value using a specific format.
ToDecimal	Converts a Double object value to a decimal value using a specific format.
ToDouble	
ToInt16	
ToInt32	

ToInt64	
ToSByte	
ToSingle	
toString	
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	

See Also

Reference

Double Class

Concepts

java.lang Package

Double.byteValue Method

Returns the byte value of a Double object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte byteValue();
```

Return Value

The byte value of the Double object.

Example

This example creates an instance of the Double class and initializes it with the value -22.1, and then converts it to a byte value.

```
// Double3.jsl
// Double constructor example

public class MyClass
{
    public static void main(String[] args)
        {
            Double d = new Double("-22.1");
            byte b = d.byteValue();

            System.out.println("The double number is: "+ d);
            System.out.println("The byte value is: "+ b);
        }
}

/*
Output:
The double number is: -22.1
The byte value is: -22
*/
```

Remarks

The byte value of a Double object doesn't include the fraction part.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.compareTo Method

Overload List

Name	Description
Double.compareTo (double)	Compares a Double object to the current object.
Double.compareTo (Object)	Compares the current Double object to another Double object.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.compareTo Method (Double)

Compares a Double object to the current object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Double adouble);
```

Parameters

adouble

The Double object to compare to.

Return Value

0 if the two objects are identical; -1 otherwise.

Example

In this example, two Double objects are created and compared to each other. The result of the comparison was false (-1).

```
// d-compareTo1.jsl
// compareTo Example

public class MyClass
{
    public static void main(String[] args)
        {
            Double d1 = new Double(-22.1);
            Double d2 = new Double(22.1);
            int i = d1.compareTo(d2);

            System.out.println("The result of the comparison: "+ i);
        }
}

/*
Output:
The result of the comparison: -1
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.compareTo Method (Object)

Compares the current Double object to another Double object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object adouble);
```

Parameters

adouble

The Double object to compare to.

Return Value

0 if the two objects are identical, 1 otherwise.

Example

In this example, two Double objects are created and compared to each other. The result of the comparison was false (1).

```
// d-compareTo2.jsl
// compareTo Example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d1 = new Double(123);
        Double d2 = new Double(12);

        System.out.print("The comparison result = "+ d1.compareTo(d2));
    }
}
/*
The comparison result = 1
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.doubleToLongBits Method

Returns the 64-bit representation of a double number as per IEEE 754 floating point standard.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static long doubleToLongBits(
    double d);
```

Parameters

d

A double expression.

Return Value

The 64-bit representation of the double number.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.doubleValue Method

Returns the double value of a Double object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double doubleValue();
```

Return Value

The double value of a Double object.

Example

In this example, a Double object is created, and then the double value of this object is retrieved and displayed.

```
// DoubleValue.jsl
// doubleValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d = new Double(123.45);

        System.out.print("The double value is: "+ d.doubleValue());
    }
}

/*
Output:
The double value is: 123.45
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.equals Method

Compares the current Double object to another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

true if the two objects are identical; false otherwise.

Example

In this example, two Double objects are created and compared to each other. The result of the comparison was true.

```
// d-equals.jsl
// equals Example

public class MyClass
{
   public static void main(String[] args)
   {
      Double d1 = new Double(123);
      Double d2 = new Double(123);

      System.out.print("The equality is: "+ d1.equals(d2));
   }
}

/*
The equality is: true
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.floatValue Method

Returns the float value of a Double object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float floatValue();
```

Return Value

The float value of the Double object.

Example

In this example, a Double object is created, and then the float value of this object is retrieved and displayed.

```
// FloatValue.jsl
// floatValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d = new Double(123.45);

        System.out.print("The float value is: "+ d.floatValue());
    }
}

/*
Output:
The float value is: 123.45
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.hashCode Method

Retrieves the hash code of the current Double object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code of the Double object.

Example

In this example, a Double object is created, and then the hash code value for this object is retrieved and displayed.

```
// HashcodeValue.jsl
// GetHashCode example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d = new Double(123.45);

        System.out.print("The hash code is: "+ d.GetHashCode());
    }
}

/*
Output:
The hash code is: -1936584703
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.intValue Method

Returns the int value of a 9e6e1c4d-6daf-46d6-8f53-819deae0168ae object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int intValue();
```

Return Value

The int value of the Doublee object.

Example

In this example, a Double object is created, and then the int value of this object is retrieved and displayed.

```
// intValue1.jsl
// intValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d = new Double(123.65);

        System.out.print("The int value is: "+ d.intValue());
    }
}

/*
Output:
The int value is: 123
*/
```

Remarks

The int value is the integer part of the floating-point number (with the fraction truncated).

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.isInfinite Method

Checks if the value of a Double object is infinite.

Overload List

Name	Description
Double.isInfinite ()	Checks if the value of a Double object is infinitely large.
Double.isInfinite (double)	Checks if a double value is infinitely large.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.isInfinite Method ()

Checks if the value of a Double object is infinitely large.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean isInfinite();
```

Return Value

true if the object is infinite; false otherwise.

Example

In this example, two Double objects are created tested if infinite. The result of the test was true for the object that contains a value divided by zero.

```
// Infinite1.jsl
// isInfinite example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d1 = new Double(123.45);
        Double d2 = new Double(123.45/0.0);

        System.out.println("Infinite object #1? "+ d1.isInfinite());
        System.out.println("Infinite object #2? "+ d2.isInfinite());

    }
}

/*
Output:
Infinite object #1? false
Infinite object #2? true
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.isInfinite Method (Double)

Checks if a double value is infinitely large.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isInfinite(
    double f);
```

Parameters

f

The double expression to be tested.

Return Value

true if the value to tested is infinity; false otherwise.

Example

In this example, you test two double expressions to see if they are infinite. The result of the test was true for one of them, which contains a value divided by zero.

```
// Infinite2.jsl
// isInfinite example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("Infinite object #1? "
                           + Double.isInfinite(1.1));
        System.out.println("Infinite object #2? "
                           + Double.isInfinite(1.1/0.0));
    }
}

/*
Output:
Infinite object #1? false
Infinite object #2? true
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.isNaN Method

Checks if the value of a Double object is "Not a Number."

Overload List

Name	Description
Double.isNaN ()	Checks if the value of a Double object is "Not a Number."
Double.isNaN (double)	Checks if double value is "Not a Number."

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.isNaN Method ()

Checks if the value of a Double object is "Not a Number."

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean isNaN();
```

Return Value

true if the value to be tested is NaN (Not a Number); false otherwise.

Example

In the following example, two Double objects are created and tested to check if they contain a NaN value. The test of one object, which contains the square root of a negative number, returned true.

```
// isNan1.jsl
// isNaN example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d1 = new Double(Math.sqrt(-1.0));
        Double d2 = new Double(Math.sqrt(16.0));

        System.out.println("NaN object #1?: "+ d1.isNaN());
        System.out.println("NaN object #2?: "+ d2.isNaN());
    }
}

/*
Output:
NaN object #1?: true
NaN object #2?: false
*/
```

Remarks

The NaN value is produced by some operations such as the square root of a negative number.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.isNaN Method (Double)

Checks if double value is "Not a Number."

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isNaN(
    double f);
```

Parameters

f

The double expression to be tested.

Return Value

true if the value to be tested is NaN (Not a Number); false otherwise.

Example

In this example, you test two double expressions to see if they evaluate to a NaN value. The test of one expression, which contains the square root of a negative number, returned true.

```
// isNan2.jsl
// isNaN example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("NaN object #1? "+
                           Double.isNaN(Math.sqrt(-1.0)));
        System.out.println("NaN object #2? "+
                           Double.isNaN(Math.sqrt(16.0)));
    }
}

/*
Output:
NaN object #1? true
NaN object #2? false
*/
```

Remarks

The NaN value is produced by some operations such as the square root of a negative number.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.longBitsToDouble Method

Returns double number that corresponds to the 64-bit representation of a double number.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double longBitsToDouble(
    long bits);
```

Parameters

bits

The 64-bit representation of a double number.

Return Value

The double number equivalent to the 64-bit representation of a double number.

Example

The following example does two conversions. First, it converts the value 1.2 to a 64-bit representation, and then converts the latter value to a double number. The print statement displays the original number 1.2.

```
// longBits1.jsl
// ongBitsToDouble example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The double value is: " +
           Double.longBitsToDouble(Double.doubleToLongBits(1.2)));
    }
}

/*
Output:
The double value is: 1.2
*/
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.longValue Method

Returns the long value of a Double object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long longValue();
```

Return Value

The long value of the Double object.

Example

In this example, a Double object is created and the long value of the object is retrieved and displayed.

```
// longValue1.jsl
// longValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d = new Double(123.45);

        System.out.println("The long value is: "+ d.longValue());
    }
}

/*
Output:
The long value is: 123
*/
```

Remarks

The long value is the integer part of the floating-point number (with the fraction truncated).

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.parseDouble Method

Converts a string to a double value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double parseDouble(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be converted.

Return Value

The double value of the string parameter.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.shortValue Method

Returns the short value of a Double object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short shortValue();
```

Return Value

The short value of the Double object.

Example

In this example, a Double object is created and the short value of the object is retrieved and displayed.

```
// shortValue1.jsl
// shortValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Double d = new Double(123.45);

        System.out.println("The short value is: "+ d.shortValue());
    }
}

/*
Output:
The short value is: 123
*/
```

Remarks

The short value is the integer part of the floating-point number (with the fraction truncated).

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToBoolean Method

Converts a Double object value to a Boolean value using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean ToBoolean(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The Boolean value of the Double object.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToByte Method

Converts a Double object value to a byte using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Byte ToByte(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The byte value of the Double object.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToChar Method

Converts a Double object value to a char using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public char ToChar(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The char value of the Double object.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToDateTime Method

Converts a Double object value to a date-time value using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.DateTime ToDateTime(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The date-time value of the Double object.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToDecimal Method

Converts a Double object value to a decimal value using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Decimal ToDecimal(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The decimal value of the Double object.

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToDouble Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double ToDouble(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short ToInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int ToInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long ToInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToSByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte ToSByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToSingle Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float ToSingle(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.toString Method

Overload List

Name	Description
Double.ToString ()	
Double.ToString (IFormatProvider)	

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.toString Method ()

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.toString Method (IFormatProvider)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String ToString(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToType Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object ToType(
    System.Type conversionType,
    System.IFormatProvider provider);
```

Parameters

conversionType

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToUInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt16 ToUInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToUInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt32 ToUInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.ToUInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt64 ToUInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Double.valueOf Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Double valueOf(
    java.lang.String s) throws java.lang.NumberFormatException;
```

Parameters

s

See Also

Reference

Double Class

Concepts

Double Members

java.lang Package

Error Class

Represents the base error for all errors in the J# Class Libraries. An error indicates a problem in the underlying execution engine.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.Error
    extends java.lang.Throwable
```

Remarks

An error is an indication of a problem in the underlying execution engine. An example might be if there was no more memory available to the application. Applications do not typically handle errors, because there usually isn't anything that can be done to resolve the problem. Compare this to exceptions, which should be handled by the application.

For more information about the differences between errors and exceptions, see Exception.

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.awt.AWTError

java.lang.LinkageError

java.lang.VirtualMachineError

See Also

Concepts

Error Members

java.lang Package

Error Members

Represents the base error for all errors in the J# Class Libraries. An error indicates a problem in the underlying execution engine.

The following tables list the members exposed by the Error type.

Public Constructors

Name	Description
Error	Overloaded. Initializes a new instance of an Error object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various errors to a java.lang.Error.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Error Class

Concepts

java.lang Package

Error Constructor

Initializes a new instance of an Error object.

Overload List

Name	Description
Error ()	Initializes a new instance of an Error object.
Error (String)	Initializes a new instance of an Error object with the given message.
Error (SerializationInfo, StreamingContext)	Initializes a new instance of an Error object during deserialization.
Error (String, Exception)	Initializes a new instance of an Error object with the given message and inner exception.

See Also

Reference

Error Class

Concepts

Error Members

java.lang Package

Error Constructor ()

Initializes a new instance of an Error object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Error();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Error Class

Concepts

Error Members

java.lang Package

Error Constructor (String)

Initializes a new instance of an Error object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Error(
    java.lang.String s);
```

Parameters

s

A message describing the error condition.

See Also

Reference

Error Class

Concepts

Error Members

java.lang Package

Error Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an Error object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Error(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

Error Class

Concepts

Error Members

java.lang Package

Error Constructor (String, Exception)

Initializes a new instance of an Error object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Error(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the error condition.

inner

An inner exception that led to the Error.

See Also

Reference

Error Class

Concepts

Error Members

java.lang Package

Error Methods

Public Methods

Name	Description
<exceptFilter>	Maps various errors to a java.lang.Error.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Error Class

Concepts

java.lang Package

Error Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

Error Class

Concepts

java.lang Package

Exception Class (J#)

Represents the base exception for all exceptions in the J# Class Libraries. An exception indicates that an application-level error has occurred.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.Exception
    extends java.lang.Throwable
```

Remarks

An exception is an indication that an error has occurred at the application level. There are two types of exception, runtime exceptions and checked exceptions. All runtime exceptions are derived from RuntimeException. A runtime exception can be thrown in a method without having to declare that exception in the throws clause of the method. All other exceptions must be declared in the throws clause of the method. Since the compiler checks this condition for all non-runtime exceptions, they are sometimes referred to as checked exceptions.

Exceptions differ from errors in that exceptions are typically indications of an error in an application, while an error is an indication of a problem in the underlying execution engine. See Error for more information about errors. Normally an application will not handle errors, but they should handle exceptions.

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

Derived Classes

See Also

Concepts

Exception Members

java.lang Package

Exception Members

Represents the base exception for all exceptions in the J# Class Libraries. An exception indicates that an application-level error has occurred.

The following tables list the members exposed by the Exception type.

Public Constructors

Name	Description
Exception	Overloaded. Initializes a new instance of an Exception object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various exceptions to a java.lang.Exception.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Exception Class

Concepts

java.lang Package

Exception Constructor

Initializes a new instance of an Exception object.

Overload List

Name	Description
Exception ()	Initializes a new instance of an Exception object.
Exception (String)	Initializes a new instance of an Exception object with the given message.
Exception (SerializationInfo, StreamingContext)	Initializes a new instance of an Exception object during deserialization.
Exception (String, Exception)	Initializes a new instance of an Exception object with the given message and inner exception.

See Also

Reference

Exception Class

Concepts

Exception Members

java.lang Package

Exception Constructor ()

Initializes a new instance of an Exception object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Exception();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Exception Class

Concepts

Exception Members

java.lang Package

Exception Constructor (String)

Initializes a new instance of an Exception object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Exception(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also

Reference

Exception Class

Concepts

Exception Members

java.lang Package

Exception Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an Exception object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Exception(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

Exception Class

Concepts

Exception Members

java.lang Package

Exception Constructor (String, Exception)

Initializes a new instance of an Exception object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Exception(
    java.lang.String msg,
    System.Exception innerEx);
```

Parameters

msg

A message describing the exceptional condition.

innerEx

An inner exception that led to the Exception.

See Also

Reference

Exception Class

Concepts

Exception Members

java.lang Package

Exception Methods

Public Methods

Name	Description
<exceptFilter>	Maps various exceptions to a java.lang.Exception.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Exception Class

Concepts

java.lang Package

Exception Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

Exception Class

Concepts

java.lang Package

ExceptionInInitializerError Class

The error that is thrown when the execution engine encounters an exception during initialization of a class from the static initializer.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.ExceptionInInitializerError
    extends java.lang.LinkageError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.ExceptionInInitializerError

See Also

Concepts

ExceptionInInitializerError Members

java.lang Package

ExceptionInInitializerError Members

The error that is thrown when the execution engine encounters an exception during initialization of a class from the static initializer.

The following tables list the members exposed by the ExceptionInInitializerError type.

Public Constructors

Name	Description
ExceptionInInitializerError	Overloaded. Initializes a new instance of an ExceptionInInitializerError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various initialization errors to a java.lang.ExceptionInInitializerError.
checkAndThrowException	Checks the exception that was thrown during initialization.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
getException	Gets the exception that was thrown during initialization.
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden. Retrieves the data to be serialized.

getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ExceptionInInitializerError Class

Concepts

java.lang Package

ExceptionInInitializerError Constructor

Initializes a new instance of an ExceptionInInitializerError object.

Overload List

Name	Description
ExceptionInInitializerError ()	Initializes a new instance of an ExceptionInInitializerError object.
ExceptionInInitializerError (String)	Initializes a new instance of an ExceptionInInitializerError object with the given message.
ExceptionInInitializerError (Throwable)	Initializes a new instance of an ExceptionInInitializerError object with the exception that was thrown during initialization.
ExceptionInInitializerError (SerializationInfo, StreamingContext)	Initializes a new instance of an ExceptionInInitializerError object during deserialization.
ExceptionInInitializerError (String, Exception)	Initializes a new instance of an ExceptionInInitializerError object with the given message and inner exception.

See Also

Reference

ExceptionInInitializerError Class

Concepts

ExceptionInInitializerError Members

java.lang Package

ExceptionInInitializerError Constructor ()

Initializes a new instance of an ExceptionInInitializerError object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ExceptionInInitializerError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ExceptionInInitializerError Class

Concepts

ExceptionInInitializerError Members

java.lang Package

ExceptionInInitializerError Constructor (String)

Initializes a new instance of an ExceptionInInitializerError object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ExceptionInInitializerError(
    java.lang.String s);
```

Parameters

s

A message describing the error condition.

See Also
Reference
ExceptionInInitializerError Class
Concepts
ExceptionInInitializerError Members
java.lang Package

ExceptionInInitializerError Constructor (Throwable)

Initializes a new instance of an ExceptionInInitializerError object with the exception that was thrown during initialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ExceptionInInitializerError(
    java.lang.Throwable thrown);
```

Parameters

thrown

The exception that was thrown during initialization.

See Also

Reference

ExceptionInInitializerError Class

Concepts

ExceptionInInitializerError Members

java.lang Package

ExceptionInInitializerError Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an ExceptionInInitializerError object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.ExceptionInInitializerError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
ExceptionInInitializerError Class
Concepts
ExceptionInInitializerError Members
java.lang Package

ExceptionInInitializerError Constructor (String, Exception)

Initializes a new instance of an ExceptionInInitializerError object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.ExceptionInInitializerError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the error condition.

inner

An inner exception that led to the ExceptionInInitializerError.

See Also

Reference

ExceptionInInitializerError Class

Concepts

ExceptionInInitializerError Members

java.lang Package

ExceptionInInitializerError Methods

Public Methods

Name	Description
<exceptFilter>	Maps various initialization errors to a java.lang.ExceptionInInitializerError.
checkAndThrowException	Checks the exception that was thrown during initialization.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
getException	Gets the exception that was thrown during initialization.
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden. Retrieves the data to be serialized.
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ExceptionInInitializerError Class

Concepts

java.lang Package

ExceptionInInitializerError.checkAndThrowException Method

Checks the exception that was thrown during initialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void checkAndThrowException(
    java.lang.Throwable thrown) throws java.lang.Throwable;
```

Parameters

thrown

The exception that was thrown during initialization.

See Also

Reference

ExceptionInInitializerError Class

Concepts

ExceptionInInitializerError Members

java.lang Package

ExceptionInInitializerError.getException Method

Gets the exception that was thrown during initialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Throwable getException();
```

Return Value

The exception that was throws during initialization.

See Also

Reference

ExceptionInInitializerError Class

Concepts

ExceptionInInitializerError Members

java.lang Package

ExceptionInInitializerError.GetObjectData Method

Retrieves the data to be serialized.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization process.

See Also

Reference

ExceptionInInitializerError Class

Concepts

ExceptionInInitializerError Members

java.lang Package

ExceptionInInitializerError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ExceptionInInitializerError Class

Concepts

java.lang Package

Float Class

The Float class wraps the primitive type float. It contains several methods for conversion between String and Float and processing Float objects.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Float
    extends java.lang.Number
    implements System.IConvertible, java.lang.Comparable
```

Inheritance Hierarchy

java.lang.Object

  java.lang.Number

   java.lang.Float

See Also

Concepts

Float Members

java.lang Package

Float Members

The Float class wraps the primitive type float. It contains several methods for conversion between String and Float and processing Float objects.

The following tables list the members exposed by the Float type.

Public Constructors

Name	Description
Float	Overloaded.

Public Fields

Name	Description
MAX_VALUE	A constant field that stores the maximum finite value of the float type.
MIN_VALUE	A constant field that stores the minimum positive value of the float type.
NaN	A constant that stores the Not-a-Nmber value of the float type.
NEGATIVE_INFINITY	A constant that stores the negative infinity value for the float type.
POSITIVE_INFINITY	A constant that stores the positive infinity value for the float type.
TYPE	The constant that stores the Class instance of the primitive type float.

Public Methods

Name	Description
byteValue	Overridden. Returns the value of the current Float object converted to a byte.
compareTo	Overloaded.
doubleValue	Overridden. Returns the double value of the current Float object.
equals	Overridden. Checks if the current Float object is equal to another specified object.
floatToIntBits	Returns the int representation of the floating-point parameter.
floatValue	Overridden. Returns the float value of the current Float object.
hashCode	Overridden. Returns the hash code of the current Float object.
getClass	(inherited from Object)
intBitsToFloat	Returns the floating-point value that corresponds to a specific bit pattern.
intValue	Overridden. Returns the value of the current Float object converted to int.
isInfinite	Overloaded.
isNaN	Overloaded.

longValue	Overridden. Returns the value of the current Float object converted to long.
clone	(inherited from Number)
parseFloat	Returns a float value specified by the string parameter.
shortValue	Overridden. Returns the value of the current Float object converted to short.
ToBoolean	
ToByte	
ToChar	
ToDateTime	
ToDecimal	
ToDouble	
ToInt16	
ToInt32	
ToInt64	
ToSByte	
ToSingle	
toString	Returns a string that represents the float parameter.
toString	Overloaded. Overridden.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Returns the Float object whose float value is represented by a specified string parameter.

See Also

Reference

Float Class

Concepts

java.lang Package

Float Fields

Public Fields

Name	Description
MAX_VALUE	A constant field that stores the maximum finite value of the float type.
MIN_VALUE	A constant field that stores the minimum positive value of the float type.
NaN	A constant that stores the Not-a-Nmber value of the float type.
NEGATIVE_INFINITY	A constant that stores the negative infinity value for the float type.
POSITIVE_INFINITY	A constant that stores the positive infinity value for the float type.
TYPE	The constant that stores the Class instance of the primitive type float.

See Also

Reference

Float Class

Concepts

java.lang Package

Float.MAX_VALUE Field

A constant field that stores the maximum finite value of the float type.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final float MAX_VALUE;
```

Example

```
// f-MAX.jsl
// Float.MAX_VALUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("The maximum finite value is: " +
                         Float.MAX_VALUE);
    }
}

/*
Output:
The maximum value is: 3.40282347E38
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.MIN_VALUE Field

A constant field that stores the minimum positive value of the float type.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final float MIN_VALUE;
```

Example

```
// f-MIN.jsl
// Float.MAX_VALUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("The minimum positive value is: " + Float.MIN_VALUE);
    }
}

/*
Output:
The minimum positive value is: 1.401298E-45
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.NaN Field

A constant that stores the Not-a-Nmber value of the float type.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final float NaN;
```

Example

```
// f-MIN.jsl
// Float.MAX_VALUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("The Not-a-Number value is: " + Float.NaN);
    }
}

/*
Output:
The Not-a-Number value is: NaN
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.NEGATIVE_INFINITY Field

A constant that stores the negative infinity value for the float type.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final float NEGATIVE_INFINITY;
```

Example

```
// f-NEGIN.jsl
// Float.NEGATIVE_INFINITY example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("The negative infinity value is: " +
                         Float.NEGATIVE_INFINITY);
    }
}

/*
Output:
The negative infinity value is: -Infinity
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.POSITIVE_INFINITY Field

A constant that stores the positive infinity value for the float type.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final float POSITIVE_INFINITY;
```

Example

```
// f-POSIN.jsl
// Float.POSITIVE_INFINITY example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("The positive infinity value is: " +
                         Float.POSITIVE_INFINITY);
    }
}

/*
Output:
The positive infinity value is: Infinity
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.TYPE Field

The constant that stores the Class instance of the primitive type float.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Class TYPE;
```

Example

```
// f-TYPE.jsl
// Float.TYPE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("The TYPE value is: " + Float.TYPE);
    }
}

/*
Output:
The TYPE value is: float
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float Constructor

Overload List

Name	Description
Float (double)	Constructs a new Float object using a double parameter.
Float (float)	Constructs a new Float object using a float parameter.
Float (String)	Constructs a new Float object using a string parameter.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float Constructor (Double)

Constructs a new Float object using a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Float(
    double d);
```

Parameters

d

A double expression.

Example

```
// f-ctor1.jsl
// Float.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float(123);
        System.out.print("The float value is: " + f);
    }
}

/*
Output:
The float value is: 123.0
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float Constructor (Single)

Constructs a new Float object using a float parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Float(
    float f);
```

Parameters

f

A float expression.

Example

```
// f-ctor2.jsl
// Float.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float(123F);
        System.out.print("The float value is: " + f);
    }
}

/*
Output:
The float value is: 123.0
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float Constructor (String)

Constructs a new Float object using a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Float(
    java.lang.String s) throws java.lang.NumberFormatException;
```

Parameters

s

A string expression.

Example

```
// f-ctor3.jsl
// Float.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float("123.25");
        System.out.print("The float value is: " + f);
    }
}

/*
Output:
The float value is: 123.25
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float Methods

Public Methods

Name	Description
byteValue	Overridden. Returns the value of the current Float object converted to a byte.
compareTo	Overloaded.
doubleValue	Overridden. Returns the double value of the current Float object.
equals	Overridden. Checks if the current Float object is equal to another specified object.
floatToIntBits	Returns the int representation of the floating-point parameter.
floatValue	Overridden. Returns the float value of the current Float object.
hashCode	Overridden. Returns the hash code of the current Float object.
getClass	(inherited from Object)
intBitsToFloat	Returns the floating-point value that corresponds to a specific bit pattern.
intValue	Overridden. Returns the value of the current Float object converted to int.
isInfinite	Overloaded.
isNaN	Overloaded.
longValue	Overridden. Returns the value of the current Float object converted to long.
clone	(inherited from Number)
parseFloat	Returns a float value specified by the string parameter.
shortValue	Overridden. Returns the value of the current Float object converted to short.
ToBoolean	
ToByte	
ToChar	
ToDateTime	
ToDecimal	
ToDouble	
ToInt16	
ToInt32	

ToInt64	
ToSByte	
ToSingle	
toString	Returns a string that represents the float parameter.
toString	Overloaded. Overridden.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Returns the Float object whose float value is represented by a specified string parameter.

See Also

Reference

Float Class

Concepts

java.lang Package

Float.byteValue Method

Returns the value of the current Float object converted to a byte.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte byteValue();
```

Return Value

The value of the current Float object converted to a byte.

Example

```
// f-bVal1.jsl
// Float.byteValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float(2.25);
        System.out.println("The byte value is: " + f.byteValue());
        System.out.println("The cast value is: " + (byte)2.25);
    }
}

/*
Output:
The byte value is: 2
The cast value is: 2
*/
```

Remarks

The fraction part is truncated from the returned value. The result is the same as when you cast a float value to a byte.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.compareTo Method

Overload List

Name	Description
Float.compareTo (Float)	Numerically compares the current Float object to another Float object.
Float.compareTo (Object)	Numerically compares the current Float object to another object.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.compareTo Method (Float)

Numerically compares the current Float object to another Float object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Float aFloat);
```

Parameters

aFloat

Non null Float object to compare to.

Return Value

0 if the two objects are numerically identical. 1 if the current object is numerically greater than aFloat. -1 if the current object is numerically less than aFloat.

Example

```
// f-comp1.jsl
// Float.compareTo example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f1 = new Float(2.25);
        Float f2 = new Float(1.44);

        System.out.println("The result of the comparison is: " +
                           f1.compareTo(f1));
        System.out.println("The result of the comparison is: " +
                           f1.compareTo(f2));
        System.out.println("The result of the comparison is: " +
                           f2.compareTo(f1));
    }
}

/*
Output:
The result of the comparison is: 0
The result of the comparison is: 1
The result of the comparison is: -1
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.compareTo Method (Object)

Numerically compares the current Float object to another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

Provided that the other object is Float, the returned value is:

0 if the two objects are numerically identical.

1 if the current object is numerically greater than aFloat.

-1 if the current object is numerically less than aFloat.

Example

```
See the example on compareTo.
```

Remarks

If the other object is a Float object, the method behaves exactly like compareTo; otherwise it throws the exception java.lang.ClassCastException.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.doubleValue Method

Returns the double value of the current Float object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double doubleValue();
```

Return Value

The double value of the current Float object.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.equals Method

Checks if the current Float object is equal to another specified object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare against.

Return Value

true if the two objects are identical; false otherwise.

Example

```
// f-equal1.jsl
// Float.Equals example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f1 = new Float(2.5);
        Float f2 = new Float(2.5);
        boolean b = f1.Equals(f2);
        System.out.print("The result is: " + b);
    }
}

/*
Output:
The result is: true
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.floatToIntBits Method

Returns the int representation of the floating-point parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int floatToIntBits(
    float f);
```

Parameters

f

A float value.

Return Value

The int representation of f.

Example

```
// f-floToIntBits.jsl
// floatToIntBits example

public class MyClass
{
    public static void main(String[] args)
    {
         float f = 2.34f;
         System.out.println("The original float value is: " + f);
         // The integer bit representation:
         int i = Float.floatToIntBits(f);
         System.out.println("The int representation is: " + i);
         // Convert it back to a float value:
         float fb = Float.intBitsToFloat(i);
         System.out.println("The converted-back float is: " + fb);
    }
}

/*
Output:
The original float value is: 2.34
The int representation is: 1075167887
The converted-back float is: 2.34
*/
```

Remarks

When you convert back the resulting integer representation using the method intBitsToFloat, it gives the same float value that was used as a parameter.

The following are special cases:

If the parameter value is NaN, the result is 0x7FC00000.

If the parameter value is POSITIVE_INFINITY, the result is 0x7F800000.

If the parameter value is NEGATIVE_INFINITY, the result is 0xFF800000.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.floatValue Method

Returns the float value of the current Float object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float floatValue();
```

Return Value

The float value of the current Float object.

Example

```
// f-fval1.jsl
// Float.floatValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float("2.25");

        System.out.println("The float value is: " + f.floatValue());
    }
}

/*
Output:
The float value is: 2.25
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.hashCode Method

Returns the hash code of the current Float object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code of the current Float object.

Example

```
// f-ghashc1.jsl
// Float.GetHashCode example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float("2.25");

        System.out.println("The hash code is: " + f.GetHashCode());
    }
}

/*
Output:
The hash code is: 1074790400
*/
```

Remarks

The hash code returned from this method is the integer representation of the value of the Float object.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.intBitsToFloat Method

Returns the floating-point value that corresponds to a specific bit pattern.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static float intBitsToFloat(
    int bits);
```

Parameters

bits

An int value.

Return Value

The floating-point value that corresponds to the specific bit pattern.

Example

// f-intBitsToFloat.jsl

```
// Float.intBitsToFloat example

public class MyClass
{
    public static void main(String[] args)
    {
        // Given a specific bit pattern (= 2.34):
        int bits = 1075167887;

        int sign     = ((bits & 0x80000000) == 0) ? 1 : -1;
        int exponent = ((bits & 0x7f800000) >> 23);
        int mantissa =  (bits & 0x007fffff);

        mantissa |= 0x00800000;
        // Calculate the result:
        float f = (float)(sign * mantissa * Math.pow(2, exponent-150));

        System.out.println("The floating-point result is: " + f);
    }
}

/*
The floating-point result is: 2.34
*/
```

See also the example on floatToIntBits.

Remarks

If the parameter value is 0x7f800000, the result is POSITIVE_INFINITY.

If the parameter value is 0xff80000, the result is NEGATIVE_INFINITY.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.intValue Method

Returns the value of the current c705743f-2a78-4a9f-a207-88dd8d160060 object converted to int.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int intValue();
```

Return Value

The value of the current Float object converted to int.

Example

```
// f-intVal1.jsl
// Float.intValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float("9.5");

        System.out.println("The int value is: " + f.intValue());
    }
}

/*
Output:
The int value is: 9
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.isInfinite Method

Overload List

Name	Description
Float.isInfinite ()	Checks if the value of the current Float object is infinity.
Float.isInfinite (float)	Checks if the value of the float parameter is infinity.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.isInfinite Method ()

Checks if the value of the current Float object is infinity.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean isInfinite();
```

Return Value

true if the value of f is infinity (positive or negative); false otherwise.

Example

```
// f-isInf1.jsl
// Float.isInfinite example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float(Float.POSITIVE_INFINITY);

        System.out.println(f.isInfinite());
    }
}

/*
Output:
true
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.isInfinite Method (Single)

Checks if the value of the float parameter is infinity.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isInfinite(
    float f);
```

Parameters

f

The float value to be tested.

Return Value

true if the value of f is infinity (positive or negative); false otherwise.

Example

```
// f-isInf2.jsl
// Float.isInfinite example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(Float.isInfinite(3.14f));
    }
}

/*
Output:
false
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.isNaN Method

Overload List

Name	Description
Float.isNaN ()	Checks if the value of the current Float object is Not-a-Number (NaN).
Float.isNaN (float)	Checks if the value of the float parameter is Not-a-Number (NaN).

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.isNaN Method ()

Checks if the value of the current Float object is Not-a-Number (NaN).

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean isNaN();
```

Return Value

true if the value is NaN; false otherwise.

Example

```
// f-isNaN1.jsl
// Float.isNaN example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float(Math.sqrt(-4));

        System.out.println(f.isNaN());
    }
}

/*
Output:
true
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.isNaN Method (Single)

Checks if the value of the float parameter is Not-a-Number (NaN).

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static boolean isNaN(
    float f);
```

Parameters

f

The float value to be tested.

Return Value

true if the value is NaN; false otherwise.

Example

```
// f-isNaN2.jsl
// Float.isNaN example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(Float.isNaN(33.4f));
    }
}

/*
Output:
false
*/
```

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.longValue Method

Returns the value of the current Float object converted to long.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long longValue();
```

Return Value

The value of the current Float object converted to long.

Example

```
// f-longV1.jsl
// Float.longValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float(3456.7);

        System.out.println("The long value is: " + f.longValue());
        System.out.println("The cast value is: " + (long)3456.7);

    }
}

/*
Output:
The long value is: 3456
The cast value is: 3456
*/
```

Remarks

The returned value is the same as when you cast a float to long.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.parseFloat Method

Returns a float value specified by the string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static float parseFloat(
    java.lang.String s) throws java.lang.NumberFormatException;
```

Parameters

s

A string expression.

Return Value

The float value specified by s.

Example

```
// f-parseF1.jsl
// Float.parseFloat example

public class MyClass
{
    public static void main(String[] args)
    {
        float f = Float.parseFloat("-2.45F");
        System.out.print("The float is: " + f);
    }
}

/*
Output:
The float is: -2.45
*/
```

Remarks

The exception java.lang.NumberFormatException is thrown if the string cannot be parsed to a float value.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.shortValue Method

Returns the value of the current Float object converted to short.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short shortValue();
```

Return Value

The value of the current Float object converted to short.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToBoolean Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean ToBoolean(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Byte ToByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToChar Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public char ToChar(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToDateTime Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.DateTime ToDateTime(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToDecimal Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Decimal ToDecimal(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToDouble Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double ToDouble(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short ToInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int ToInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long ToInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToSByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte ToSByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToSingle Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float ToSingle(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.toString Method (J#)

Returns a string that represents the float parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toString(
    float f);
```

Parameters

f

A float expression.

Return Value

The string that represents f.

Example

```
// f-toStr1.jsl
// Float.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        // Convert the float to a string:
        String s = Float.toString(-3.25f);
        // Then catenate it with another string:
        s = "The string is: " + s;
        System.out.print(s);
    }
}

/*
Output:
The string is: -3.25
*/
```

Remarks

If f is NaN the result is NaN.

If f is POSITIVE_INFINITY, the result is Infinity.

If f is NEGATIVE_INFINITY, the result is -Infinity.

If f is 0.0f, the result is 0.0.

If f is -0.0f, the result is -0.0.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.toString Method

Overload List

Name	Description
Float.ToString ()	Returns a string that represents the value of the current Float object.
Float.ToString (IFormatProvider)	

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.toString Method ()

Returns a string that represents the value of the current Float object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The string that represents the value of the current Float object.

Example

```
// f-toStr2.jsl
// Float.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = new Float(2.33f);
        String s = f.toString();
        // Catenate it with another string:
        s = "The string is: " + s;
        System.out.print(s);
    }
}

/*
Output:
The string is: 2.33
*/
```

Remarks

If f is NaN the result is NaN.

If f is POSITIVE_INFINITY, the result is Infinity.

If f is NEGATIVE_INFINITY, the result is -Infinity.

If f is 0.0f, the result is 0.0.

If f is -0.0f, the result is -0.0.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.toString Method (IFormatProvider)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String ToString(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToType Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object ToType(
    System.Type conversionType,
    System.IFormatProvider provider);
```

Parameters

conversionType

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToUInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt16 ToUInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToUInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt32 ToUInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.ToUInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt64 ToUInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

Float.valueOf Method

Returns the Float object whose float value is represented by a specified string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Float valueOf(
    java.lang.String s) throws java.lang.NumberFormatException;
```

Parameters

s

The string that represents the float value.

Return Value

The Float object whose float value is represented by s.

Example

```
// f-valOf1.jsl
// Float.valueOf example

public class MyClass
{
    public static void main(String[] args)
    {
        Float f = Float.valueOf("2.25");
        System.out.print("The value is: " + f);
    }
}

/*
Output:
The value is: 2.25
*/
```

Remarks

The exception java.lang.NumberFormatException is thrown if the string cannot be parsed as a float value.

The exception java.lang.NullPointerException is thrown if the string is null.

See Also

Reference

Float Class

Concepts

Float Members

java.lang Package

IllegalAccessError Class

The error that is thrown when the execution engine attempts to access a member that it does not have permission to access.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.IllegalAccessError
    extends java.lang.IncompatibleClassChangeError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.IncompatibleClassChangeError

java.lang.IllegalAccessError

See Also

Concepts

IllegalAccessError Members

java.lang Package

IllegalAccessError Members

The error that is thrown when the execution engine attempts to access a member that it does not have permission to access.

The following tables list the members exposed by the IllegalAccessError type.

Public Constructors

Name	Description
IllegalAccessError	Overloaded. Initializes a new instance of an IllegalAccessError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various illegal access errors to a java.lang.IllegalAccessError.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalAccessError Class

Concepts

java.lang Package

IllegalAccessError Constructor

Initializes a new instance of an IllegalAccessError object.

Overload List

Name	Description
IllegalAccessError ()	Initializes a new instance of an IllegalAccessError object.
IllegalAccessError (String)	Initializes a new instance of an IllegalAccessError object with the given message.
IllegalAccessError (SerializationInfo, StreamingContext)	Initializes a new instance of an IllegalAccessError object during deserialization.
IllegalAccessError (String, Exception)	Initializes a new instance of an IllegalAccessError object with the given message and inner exception.

See Also

Reference

IllegalAccessError Class

Concepts

IllegalAccessError Members

java.lang Package

IllegalAccessError Constructor ()

Initializes a new instance of an IllegalAccessError object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalAccessError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

IllegalAccessError Class

Concepts

IllegalAccessError Members

java.lang Package

IllegalAccessError Constructor (String)

Initializes a new instance of an IllegalAccessError object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalAccessError(
    java.lang.String s);
```

Parameters

s

A message describing the error condition.

See Also
Reference
IllegalAccessError Class
Concepts
IllegalAccessError Members
java.lang Package

IllegalAccessError Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an IllegalAccessError object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.IllegalAccessError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
IllegalAccessError Class
Concepts
IllegalAccessError Members
java.lang Package

IllegalAccessError Constructor (String, Exception)

Initializes a new instance of an IllegalAccessError object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalAccessError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the error condition.

inner

An inner exception that led to the IllegalAccessError.

See Also

Reference

IllegalAccessError Class

Concepts

IllegalAccessError Members

java.lang Package

IllegalAccessError Methods

Public Methods

Name	Description
<exceptFilter>	Maps various illegal access errors to a java.lang.IllegalAccessError.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalAccessError Class

Concepts

java.lang Package

IllegalAccessError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

IllegalAccessError Class

Concepts

java.lang Package

IllegalAccessException Class

The exception that is thrown when attempting to access a member during reflection that the caller does not have permission to access.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.IllegalAccessException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.IllegalAccessException

See Also

Concepts

IllegalAccessException Members

java.lang Package

IllegalAccessException Members

The exception that is thrown when attempting to access a member during reflection that the caller does not have permission to access.

The following tables list the members exposed by the IllegalAccessException type.

Public Constructors

Name	Description
IllegalAccessException	Overloaded. Initializes a new instance of an IllegalAccessException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalAccessException Class

Concepts

java.lang Package

IllegalAccessException Constructor

Initializes a new instance of an IllegalAccessException object.

Overload List

Name	Description
IllegalAccessException ()	Initializes a new instance of an IllegalAccessException object.
IllegalAccessException (String)	Initializes a new instance of an IllegalAccessException object with the given message.
IllegalAccessException (SerializationInfo, StreamingContext)	Initializes a new instance of an IllegalAccessException object during deserialization.
IllegalAccessException (String, Exception)	Initializes a new instance of an IllegalAccessException object with the message and inner exception.

See Also

Reference

IllegalAccessException Class

Concepts

IllegalAccessException Members

java.lang Package

IllegalAccessException Constructor ()

Initializes a new instance of an IllegalAccessException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalAccessException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

IllegalAccessException Class

Concepts

IllegalAccessException Members

java.lang Package

IllegalAccessException Constructor (String)

Initializes a new instance of an IllegalAccessException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalAccessException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also

Reference

IllegalAccessException Class

Concepts

IllegalAccessException Members

java.lang Package

IllegalAccessException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an IllegalAccessException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.IllegalAccessException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
IllegalAccessException Class
Concepts
IllegalAccessException Members
java.lang Package

IllegalAccessException Constructor (String, Exception)

Initializes a new instance of an IllegalAccessException object with the message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalAccessException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the IllegalAccessException.

See Also

Reference

IllegalAccessException Class

Concepts

IllegalAccessException Members

java.lang Package

IllegalAccessException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalAccessException Class

Concepts

java.lang Package

IllegalAccessException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

IllegalAccessException Class

Concepts

java.lang Package

IllegalArgumentException Class

The exception that is thrown when an invalid argument is provided to a method call.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.IllegalArgumentException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.IllegalArgumentException

java.lang.IllegalThreadStateException

java.lang.NumberFormatException

java.security.InvalidParameterException

See Also

Concepts

IllegalArgumentException Members

java.lang Package

IllegalArgumentException Members

The exception that is thrown when an invalid argument is provided to a method call.

The following tables list the members exposed by the IllegalArgumentException type.

Public Constructors

Name	Description
IllegalArgumentException	Overloaded. Initializes a new instance of an IllegalArgumentException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various illegal argument exceptions to a java.lang.IllegalArgumentException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalArgumentException Class

Concepts

java.lang Package

IllegalArgumentException Constructor

Initializes a new instance of an IllegalArgumentException object.

Overload List

Name	Description
IllegalArgumentException ()	Initializes a new instance of an IllegalArgumentException object.
IllegalArgumentException (String)	Initializes a new instance of an IllegalArgumentException object with the given message.
IllegalArgumentException (SerializationInfo, StreamingContext)	Initializes a new instance of an IllegalArgumentException during deserialization.
IllegalArgumentException (String, Exception)	Initializes a new instance of an IllegalArgumentException with the given message and inner exception.

See Also

Reference

IllegalArgumentException Class

Concepts

IllegalArgumentException Members

java.lang Package

IllegalArgumentException Constructor ()

Initializes a new instance of an IllegalArgumentException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalArgumentException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

IllegalArgumentException Class

Concepts

IllegalArgumentException Members

java.lang Package

IllegalArgumentException Constructor (String)

Initializes a new instance of an IllegalArgumentException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalArgumentException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
IllegalArgumentException Class
Concepts
IllegalArgumentException Members
java.lang Package

IllegalArgumentException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an IllegalArgumentException during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.IllegalArgumentException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
IllegalArgumentException Class
Concepts
IllegalArgumentException Members
java.lang Package

IllegalArgumentException Constructor (String, Exception)

Initializes a new instance of an IllegalArgumentException with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalArgumentException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the IllegalArgumentException.

See Also
Reference
IllegalArgumentException Class
Concepts
IllegalArgumentException Members
java.lang Package

IllegalArgumentException Methods

Public Methods

Name	Description
<exceptFilter>	Maps various illegal argument exceptions to a java.lang.IllegalArgumentException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalArgumentException Class

Concepts

java.lang Package

IllegalArgumentException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

IllegalArgumentException Class

Concepts

java.lang Package

IllegalMonitorStateException Class

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.IllegalMonitorStateException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.IllegalMonitorStateException

See Also

Concepts

IllegalMonitorStateException Members

java.lang Package

IllegalMonitorStateException Members

The following tables list the members exposed by the IllegalMonitorStateException type.

Public Constructors

Name	Description
IllegalMonitorStateException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalMonitorStateException Class

Concepts

java.lang Package

IllegalMonitorStateException Constructor

Overload List

Name	Description
IllegalMonitorStateException ()	
IllegalMonitorStateException (String)	
IllegalMonitorStateException (SerializationInfo, StreamingContext)	
IllegalMonitorStateException (String, Exception)	

See Also

Reference

IllegalMonitorStateException Class

Concepts

IllegalMonitorStateException Members

java.lang Package

IllegalMonitorStateException Constructor ()

Initializes a new instance of the IllegalMonitorStateException Class .

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalMonitorStateException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

IllegalMonitorStateException Class

Concepts

IllegalMonitorStateException Members

java.lang Package

IllegalMonitorStateException Constructor (String)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalMonitorStateException(
    java.lang.String s);
```

Parameters

s

See Also
Reference
IllegalMonitorStateException Class
Concepts
IllegalMonitorStateException Members
java.lang Package

IllegalMonitorStateException Constructor (SerializationInfo, StreamingContext)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.IllegalMonitorStateException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also
Reference
IllegalMonitorStateException Class
Concepts
IllegalMonitorStateException Members
java.lang Package

IllegalMonitorStateException Constructor (String, Exception)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalMonitorStateException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also
Reference
IllegalMonitorStateException Class
Concepts
IllegalMonitorStateException Members
java.lang Package

IllegalMonitorStateException Methods

Public Methods

Name	Description
<exceptFilter>	
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalMonitorStateException Class

Concepts

java.lang Package

IllegalMonitorStateException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

IllegalMonitorStateException Class

Concepts

java.lang Package

IllegalStateException Class

The exception that is thrown when the program has entered an invalid state.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.IllegalStateException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.IllegalStateException

java.awt.IllegalComponentStateException

See Also

Concepts

IllegalStateException Members

java.lang Package

IllegalStateException Members

The exception that is thrown when the program has entered an invalid state.

The following tables list the members exposed by the IllegalStateException type.

Public Constructors

Name	Description
IllegalStateException	Overloaded. Initializes a new instance of an IllegalStateException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalStateException Class

Concepts

java.lang Package

IllegalStateException Constructor

Initializes a new instance of an IllegalStateException object.

Overload List

Name	Description
IllegalStateException ()	Initializes a new instance of an IllegalStateException object.
IllegalStateException (String)	Initializes a new instance of an IllegalStateException object with the given message.
IllegalStateException (SerializationInfo, StreamingContext)	Initializes a new instance of an IllegalStateException during deserialization.
IllegalStateException (String, Exception)	Initializes a new instance of an IllegalStateException object with the given message and inner exception.

See Also

Reference

IllegalStateException Class

Concepts

IllegalStateException Members

java.lang Package

IllegalStateException Constructor ()

Initializes a new instance of an IllegalStateException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalStateException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

IllegalStateException Class

Concepts

IllegalStateException Members

java.lang Package

IllegalStateException Constructor (String)

Initializes a new instance of an IllegalStateException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalStateException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
IllegalStateException Class
Concepts
IllegalStateException Members
java.lang Package

IllegalStateException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an IllegalStateException during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.IllegalStateException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
IllegalStateException Class
Concepts
IllegalStateException Members
java.lang Package

IllegalStateException Constructor (String, Exception)

Initializes a new instance of an IllegalStateException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalStateException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the IllegalStateException.

See Also

Reference

IllegalStateException Class

Concepts

IllegalStateException Members

java.lang Package

IllegalStateException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalStateException Class

Concepts

java.lang Package

IllegalStateException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

IllegalStateException Class

Concepts

java.lang Package

IllegalThreadStateException Class

The exception that is thrown when a thread has entered an illegal state.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.IllegalThreadStateException
    extends java.lang.IllegalArgumentException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.IllegalArgumentException

java.lang.IllegalThreadStateException

See Also

Concepts

IllegalThreadStateException Members

java.lang Package

IllegalThreadStateException Members

The exception that is thrown when a thread has entered an illegal state.

The following tables list the members exposed by the IllegalThreadStateException type.

Public Constructors

Name	Description
IllegalThreadStateException	Overloaded. Initializes a new instance of an IllegalThreadStateException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Do not document.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalThreadStateException Class

Concepts

java.lang Package

IllegalThreadStateException Constructor

Initializes a new instance of an IllegalThreadStateException object.

Overload List

Name	Description
IllegalThreadStateException ()	Initializes a new instance of an IllegalThreadStateException object.
IllegalThreadStateException (String)	Initializes a new instance of an IllegalThreadStateException object with the given message.
IllegalThreadStateException (SerializationInfo, StreamingContext)	Initializes a new instance of an IllegalThreadStateException object during deserialization.
IllegalThreadStateException (String, Exception)	Initializes a new instance of an IllegalThreadStateException object with the given message and inner exception.

See Also

Reference

IllegalThreadStateException Class

Concepts

IllegalThreadStateException Members

java.lang Package

IllegalThreadStateException Constructor ()

Initializes a new instance of an IllegalThreadStateException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalThreadStateException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

IllegalThreadStateException Class

Concepts

IllegalThreadStateException Members

java.lang Package

IllegalThreadStateException Constructor (String)

Initializes a new instance of an IllegalThreadStateException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalThreadStateException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
IllegalThreadStateException Class
Concepts
IllegalThreadStateException Members
java.lang Package

IllegalThreadStateException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an IllegalThreadStateException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.IllegalThreadStateException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

IllegalThreadStateException Class

Concepts

IllegalThreadStateException Members

java.lang Package

IllegalThreadStateException Constructor (String, Exception)

Initializes a new instance of an IllegalThreadStateException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IllegalThreadStateException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the IllegalThreadStateException being thrown.

See Also

Reference

IllegalThreadStateException Class

Concepts

IllegalThreadStateException Members

java.lang Package

IllegalThreadStateException Methods

Public Methods

Name	Description
<exceptFilter>	Do not document.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IllegalThreadStateException Class

Concepts

java.lang Package

IllegalThreadStateException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

IllegalThreadStateException Class

Concepts

java.lang Package

IncompatibleClassChangeError Class

The error that is thrown when an incompatible change is made to a class.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.IncompatibleClassChangeError
    extends java.lang.LinkageError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.IncompatibleClassChangeError

java.lang.AbstractMethodError

java.lang.IllegalAccessError

java.lang.InstantiationError

java.lang.NoSuchFieldError

java.lang.NoSuchMethodError

See Also

Concepts

IncompatibleClassChangeError Members

java.lang Package

IncompatibleClassChangeError Members

The error that is thrown when an incompatible change is made to a class.

The following tables list the members exposed by the IncompatibleClassChangeError type.

Public Constructors

Name	Description
IncompatibleClassChangeError	Overloaded. Initializes a new instance of an IncompatibleClassChangeError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Do not document.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IncompatibleClassChangeError Class

Concepts

java.lang Package

IncompatibleClassChangeError Constructor

Initializes a new instance of an IncompatibleClassChangeError object.

Overload List

Name	Description
IncompatibleClassChangeError ()	Initializes a new instance of an IncompatibleClassChangeError object.
IncompatibleClassChangeError (String)	Initializes a new instance of an IncompatibleClassChangeError object with the given message.
IncompatibleClassChangeError (SerializationInfo, StreamingContext)	Initializes a new instance of an IncompatibleClassChangeError object during deserialization.
IncompatibleClassChangeError (String, Exception)	Initializes a new instance of an IncompatibleClassChangeError object with the given message and inner exception.

See Also

Reference

IncompatibleClassChangeError Class

Concepts

IncompatibleClassChangeError Members

java.lang Package

IncompatibleClassChangeError Constructor ()

Initializes a new instance of an IncompatibleClassChangeError object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IncompatibleClassChangeError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

IncompatibleClassChangeError Class

Concepts

IncompatibleClassChangeError Members

java.lang Package

IncompatibleClassChangeError Constructor (String)

Initializes a new instance of an IncompatibleClassChangeError object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IncompatibleClassChangeError(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
IncompatibleClassChangeError Class
Concepts
IncompatibleClassChangeError Members
java.lang Package

IncompatibleClassChangeError Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an IncompatibleClassChangeError object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.IncompatibleClassChangeError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
IncompatibleClassChangeError Class
Concepts
IncompatibleClassChangeError Members
java.lang Package

IncompatibleClassChangeError Constructor (String, Exception)

Initializes a new instance of an IncompatibleClassChangeError object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IncompatibleClassChangeError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the error condition.

inner

An inner exception that led to the IncompatibleClassChangeError being thrown.

See Also

Reference

IncompatibleClassChangeError Class

Concepts

IncompatibleClassChangeError Members

java.lang Package

IncompatibleClassChangeError Methods

Public Methods

Name	Description
<exceptFilter>	Do not document.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IncompatibleClassChangeError Class

Concepts

java.lang Package

IncompatibleClassChangeError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

IncompatibleClassChangeError Class

Concepts

java.lang Package

IndexOutOfBoundsException Class

The exception that is thrown when an attempt is made to read beyond the bounds of a collection.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.IndexOutOfBoundsException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.IndexOutOfBoundsException

java.lang.ArrayIndexOutOfBoundsException

java.lang.StringIndexOutOfBoundsException

See Also

Concepts

IndexOutOfBoundsException Members

java.lang Package

IndexOutOfBoundsException Members

The exception that is thrown when an attempt is made to read beyond the bounds of a collection.

The following tables list the members exposed by the IndexOutOfBoundsException type.

Public Constructors

Name	Description
IndexOutOfBoundsException	Overloaded. Initializes a new instance of an IndexOutOfBoundsException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps various out of bounds exceptions to a java.lang.IndexOutOfBoundsException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IndexOutOfBoundsException Class

Concepts

java.lang Package

IndexOutOfBoundsException Constructor

Initializes a new instance of an IndexOutOfBoundsException object.

Overload List

Name	Description
IndexOutOfBoundsException ()	Initializes a new instance of an IndexOutOfBoundsException object.
IndexOutOfBoundsException (String)	Initializes a new instance of an IndexOutOfBoundsException object with the given message.
IndexOutOfBoundsException (SerializationInfo, StreamingContext)	Initializes a new instance of an IndexOutOfBoundsException object during deserialization.
IndexOutOfBoundsException (String, Exception)	Initializes a new instance of an IndexOutOfBoundsException object with the given message and inner exception.

See Also

Reference

IndexOutOfBoundsException Class

Concepts

IndexOutOfBoundsException Members

java.lang Package

IndexOutOfBoundsException Constructor ()

Initializes a new instance of an IndexOutOfBoundsException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IndexOutOfBoundsException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

IndexOutOfBoundsException Class

Concepts

IndexOutOfBoundsException Members

java.lang Package

IndexOutOfBoundsException Constructor (String)

Initializes a new instance of an IndexOutOfBoundsException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IndexOutOfBoundsException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also

Reference

IndexOutOfBoundsException Class

Concepts

IndexOutOfBoundsException Members

java.lang Package

IndexOutOfBoundsException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an IndexOutOfBoundsException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.IndexOutOfBoundsException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
IndexOutOfBoundsException Class
Concepts
IndexOutOfBoundsException Members
java.lang Package

IndexOutOfBoundsException Constructor (String, Exception)

Initializes a new instance of an IndexOutOfBoundsException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.IndexOutOfBoundsException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the IndexOutOfBoundsException.

See Also
Reference
IndexOutOfBoundsException Class
Concepts
IndexOutOfBoundsException Members
java.lang Package

IndexOutOfBoundsException Methods

Public Methods

Name	Description
<exceptFilter>	Maps various out of bounds exceptions to a java.lang.IndexOutOfBoundsException.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

IndexOutOfBoundsException Class

Concepts

java.lang Package

IndexOutOfBoundsException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

IndexOutOfBoundsException Class

Concepts

java.lang Package

InstantiationError Class

The error that is thrown when the execution engine encounters a problem instantiating a new object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.InstantiationError
    extends java.lang.IncompatibleClassChangeError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.IncompatibleClassChangeError

java.lang.InstantiationError

See Also

Concepts

InstantiationError Members

java.lang Package

InstantiationError Members

The error that is thrown when the execution engine encounters a problem instantiating a new object.

The following tables list the members exposed by the InstantiationError type.

Public Constructors

Name	Description
InstantiationError	Overloaded. Initializes a new instance of an InstantiationError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InstantiationError Class

Concepts

java.lang Package

InstantiationError Constructor

Initializes a new instance of an InstantiationError object.

Overload List

Name	Description
InstantiationError ()	Initializes a new instance of an InstantiationError object.
InstantiationError (String)	Initializes a new instance of an InstantiationError object with the given message.
InstantiationError (SerializationInfo, StreamingContext)	Initializes a new instance of an InstantiationError object during deserialization.
InstantiationError (String, Exception)	Initializes a new instance of an InstantiationError object with the given message and inner exception.

See Also

Reference

InstantiationError Class

Concepts

InstantiationError Members

java.lang Package

InstantiationError Constructor ()

Initializes a new instance of an InstantiationError object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InstantiationError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

InstantiationError Class

Concepts

InstantiationError Members

java.lang Package

InstantiationError Constructor (String)

Initializes a new instance of an InstantiationError object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InstantiationError(
    java.lang.String s);
```

Parameters

s

A message describing the error condition.

See Also
Reference
InstantiationError Class
Concepts
InstantiationError Members
java.lang Package

InstantiationError Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an InstantiationError object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.InstantiationError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

InstantiationError Class

Concepts

InstantiationError Members

java.lang Package

InstantiationError Constructor (String, Exception)

Initializes a new instance of an InstantiationError object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InstantiationError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the error condition.

inner

An inner exception that led to the InstantiationError.

See Also

Reference

InstantiationError Class

Concepts

InstantiationError Members

java.lang Package

InstantiationError Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InstantiationError Class

Concepts

java.lang Package

InstantiationError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

InstantiationError Class

Concepts

java.lang Package

InstantiationException Class

The exception that is thrown when an error occurs instantiating (creating) an object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.InstantiationException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.InstantiationException

See Also

Concepts

InstantiationException Members

java.lang Package

InstantiationException Members

The exception that is thrown when an error occurs instantiating (creating) an object.

The following tables list the members exposed by the InstantiationException type.

Public Constructors

Name	Description
InstantiationException	Overloaded. Initializes a new instance of an InstantiationException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InstantiationException Class

Concepts

java.lang Package

InstantiationException Constructor

Initializes a new instance of an InstantiationException object.

Overload List

Name	Description
InstantiationException ()	Initializes a new instance of an InstantiationException object.
InstantiationException (String)	Initializes a new instance of an InstantiationException object with the given message.
InstantiationException (SerializationInfo, StreamingContext)	Initializes a new instance of an InstantiationException object during deserialization.
InstantiationException (String, Exception)	Initializes a new instance of an InstantiationException object with the given message and inner exception.

See Also

Reference

InstantiationException Class

Concepts

InstantiationException Members

java.lang Package

InstantiationException Constructor ()

Initializes a new instance of an InstantiationException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InstantiationException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

InstantiationException Class

Concepts

InstantiationException Members

java.lang Package

InstantiationException Constructor (String)

Initializes a new instance of an InstantiationException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InstantiationException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
InstantiationException Class
Concepts
InstantiationException Members
java.lang Package

InstantiationException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an InstantiationException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.InstantiationException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

InstantiationException Class

Concepts

InstantiationException Members

java.lang Package

InstantiationException Constructor (String, Exception)

Initializes a new instance of an InstantiationException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InstantiationException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the InstantiationException.

See Also

Reference

InstantiationException Class

Concepts

InstantiationException Members

java.lang Package

InstantiationException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InstantiationException Class

Concepts

java.lang Package

InstantiationException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

InstantiationException Class

Concepts

java.lang Package

Integer Class

Declares an Integer class type that wraps the primitive type int into an object. The class contains method members to perform tests and to convert from int to other types and vice versa.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Integer
    extends java.lang.Number
    implements System.IConvertible, java.lang.Comparable
```

Inheritance Hierarchy

java.lang.Object

  java.lang.Number

   java.lang.Integer

See Also

Concepts

Integer Members

java.lang Package

Integer Members

Declares an Integer class type that wraps the primitive type int into an object. The class contains method members to perform tests and to convert from int to other types and vice versa.

The following tables list the members exposed by the Integer type.

Public Constructors

Name	Description
Integer	Overloaded. Constructs a new Integer object.

Public Fields

Name	Description
MAX_VALUE	The maximum value an Integer object can store.
MIN_VALUE	The minimum value an Integer object can store.
TYPE	The instance of the Class type that represents the primitive type of the Integer object.

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value of an Integer object.
compareTo	Overloaded. Compares two Integer objects.
decode	Decodes a string and returns the corresponding integer.
doubleValue	Overridden. Returns the double value of an Integer object.
equals	Overridden. Compares the current object to another object and returns a Boolean value.
floatValue	Overridden. Returns the float value of an Integer object.
hashCode	Overridden. Retrieves the hash code of an Integer object.
getInteger	Overloaded. Returns the integer value of a property.
getClass	(inherited from Object)
intValue	Overridden. Returns the int value of the current Integer object.
longValue	Overridden. Returns the long value of the current Integer object.
clone	(inherited from Number)
parseInt	Overloaded. Parses the string parameter and converts it to an integer value.
shortValue	Overridden. Returns the short value of an Integer object.
toBinaryString	Converts the value of an integer to a binary string.

ToBoolean	Converts the value of an integer to Boolean using a specific format.
ToByte	Converts an Integer value to a byte using a specific format.
ToChar	Converts an Integer value to a char using a specific format.
ToDateTime	Converts an Integer object value to a date-time value using a specific format.
ToDecimal	Converts an Integer object value to a decimal value using a specific format.
ToDouble	Converts an Integer object value to a double value using a specific format.
toHexString	Returns the hexadecimal string representation of an int.
ToInt16	Converts an Integer object value to a 2-byte integer using a specific format.
ToInt32	Converts an Integer object value to a 4-byte integer using a specific format.
ToInt64	Converts an Integer object value to an 8-byte integer using a specific format.
toOctalString	Returns the octal number string representation of an int.
ToSByte	Converts an Integer object value to an sbyte value using a specific format.
ToSingle	Converts an Integer object value to a float value using a specific format.
toString	Overloaded.
toString	Overloaded. Overridden. Returns the string representation of the value of an Integer object.
ToType	Converts an Integer object to a specific type using a specified format.
ToUInt16	Converts an Integer object to an unsigned short integer using a specified format.
ToUInt32	Converts an Integer object to an unsigned integer using a specified format.
ToUInt64	Converts an Integer object to an unsigned long integer using a specified format.
valueOf	Overloaded.

See Also

Reference

Integer Class

Concepts

java.lang Package

Integer Fields

Public Fields

Name	Description
MAX_VALUE	The maximum value an Integer object can store.
MIN_VALUE	The minimum value an Integer object can store.
TYPE	The instance of the Class type that represents the primitive type of the Integer object.

See Also

Reference

Integer Class

Concepts

java.lang Package

Integer.MAX_VALUE Field

The maximum value an Integer object can store.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final int MAX_VALUE;
```

Example

```
// i-MaxMin.jsl
// Max and Min example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer max = new Integer(Integer.MAX_VALUE);
        Integer min = new Integer(Integer.MIN_VALUE);

        System.out.println("Max value = " + max);
        System.out.println("Min value = " + min);
    }
}

/*
Output:
Max value = 2147483647
Min value = -2147483648
*/
```

Remarks

The value of MAX_VALUE is 2147483647.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.MIN_VALUE Field

The minimum value an Integer object can store.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final int MIN_VALUE;
```

Example

See the example under MAX_VALUE.

Remarks

The value of MIN_VALUE is -2147483648.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.TYPE Field

The instance of the Class type that represents the primitive type of the Integer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Class TYPE;
```

Example

This example prints the value of Integer.TYPE.

```
// i-TYPE1.jsl
// Class TYPE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The primitive type is: " + Integer.TYPE);
    }
}

/*
Output:
The primitive type is: int
*/
```

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer Constructor

Constructs a new Integer object.

Overload List

Name	Description
Integer (int)	Constructs an Integer object that represents the integer passed as a parameter.
Integer (String)	Constructs an Integer object that represents the string passed as a parameter.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer Constructor (Int32)

Constructs an Integer object that represents the integer passed as a parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Integer(
    int value);
```

Parameters

value

Expression of the type int.

Example

This example creates an instance of the Integer class and initializes it with the value 123.

```
// Integer1.jsl
// Integer constructor example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer i = new Integer(123);

        System.out.print("The integer number is: "+ i);
    }
}

/*
Output:
The integer number is: 123
*/
```

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer Constructor (String)

Constructs an Integer object that represents the string passed as a parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Integer(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

A string parameter.

Example

This example creates an instance of the Integer class and initializes it with the string "123".

```
// Integer2.jsl
// Integer constructor example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer i = new Integer("123");

        System.out.print("The integer number is: "+ i);
    }
}

/*
Output:
The integer number is: 123
*/
```

Remarks

The string parameter should contain an int expression.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer Methods

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value of an Integer object.
compareTo	Overloaded. Compares two Integer objects.
decode	Decodes a string and returns the corresponding integer.
doubleValue	Overridden. Returns the double value of an Integer object.
equals	Overridden. Compares the current object to another object and returns a Boolean value.
floatValue	Overridden. Returns the float value of an Integer object.
hashCode	Overridden. Retrieves the hash code of an Integer object.
getInteger	Overloaded. Returns the integer value of a property.
getClass	(inherited from Object)
intValue	Overridden. Returns the int value of the current Integer object.
longValue	Overridden. Returns the long value of the current Integer object.
clone	(inherited from Number)
parseInt	Overloaded. Parses the string parameter and converts it to an integer value.
shortValue	Overridden. Returns the short value of an Integer object.
toBinaryString	Converts the value of an integer to a binary string.
ToBoolean	Converts the value of an integer to Boolean using a specific format.
ToByte	Converts an Integer value to a byte using a specific format.
ToChar	Converts an Integer value to a char using a specific format.
ToDateTime	Converts an Integer object value to a date-time value using a specific format.
ToDecimal	Converts an Integer object value to a decimal value using a specific format.
ToDouble	Converts an Integer object value to a double value using a specific format.
toHexString	Returns the hexadecimal string representation of an int.
ToInt16	Converts an Integer object value to a 2-byte integer using a specific format.
ToInt32	Converts an Integer object value to a 4-byte integer using a specific format.

ToInt64	Converts an Integer object value to an 8-byte integer using a specific format.
toOctalString	Returns the octal number string representation of an int.
ToSByte	Converts an Integer object value to an sbyte value using a specific format.
ToSingle	Converts an Integer object value to a float value using a specific format.
toString	Overloaded.
toString	Overloaded. Overridden. Returns the string representation of the value of an Integer object.
ToType	Converts an Integer object to a specific type using a specified format.
ToUInt16	Converts an Integer object to an unsigned short integer using a specified format.
ToUInt32	Converts an Integer object to an unsigned integer using a specified format.
ToUInt64	Converts an Integer object to an unsigned long integer using a specified format.
valueOf	Overloaded.

See Also

Reference

Integer Class

Concepts

java.lang Package

Integer.byteValue Method

Returns the byte value of an Integer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte byteValue();
```

Return Value

The byte value of the Integer object.

Example

```
// i-byteVal1.jsl
// compareTo Example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer i = new Integer(123);

        System.out.println("The byte value is: " + i.byteValue());
   }
}
/*
Output:
The byte value is: 123
*/
```

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.compareTo Method

Compares two Integer objects.

Overload List

Name	Description
Integer.compareTo (Integer)	Compares two Integer objects and returns an int value that represents the result of the comparison.
Integer.compareTo (Object)	Compares the Integer current object to another object, which may or may not be an Integer. In the later case it throws an exception.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.compareTo Method (Integer)

Compares two Integer objects and returns an int value that represents the result of the comparison.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Integer anInt);
```

Parameters

anInt

The Integer object to compare to.

Return Value

0 if the two objects are identical. -1 if the parameter is greater than the object. 1 if the parameter is less than the object.

Example

In this example, two Integer objects are created and compared. Depending on the value of the parameter compared to the value of the object the returned value take one of three values -1, 0, and 1.

```
// i-compareTo1.jsl
// compareTo Example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer i1 = new Integer(125);
        Integer i2 = new Integer(123);

        System.out.println("The comparison result = "+
                           i1.compareTo(i2));   // 1
        System.out.println("The comparison result = "+
                           i2.compareTo(i1));   // -1
        System.out.println("The comparison result = "+
                           i1.compareTo(i1));   // 0
    }
}
/*
Output:
The comparison result = 1
The comparison result = -1
The comparison result = 0
*/
```

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.compareTo Method (Object)

Compares the Integer current object to another object, which may or may not be an Integer. In the later case it throws an exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object anInt);
```

Parameters

anInt

The object to compare to.

Return Value

0 if the two objects are identical. -1 if the parameter is greater than the object. 1 if the parameter is less than the object.

Example

In this example, two Integer objects are created and compared. Depending on the value of the parameter compared to the value of the object the returned value take one of three values -1, 0, and 1.

A fourth comparison is made between an Integer and a Double object, but the statement is commented. If you uncomment the statement it will throw an exception.

```
// i-compareTo2.jsl
// compareTo Example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer i1 = new Integer(125);
        Integer i2 = new Integer(123);
        Double i3 = new Double(12.3);

        System.out.println("The comparison result = "+
                           i1.compareTo(i2));   // 1
        System.out.println("The comparison result = "+
                           i2.compareTo(i1));   // -1
        System.out.println("The comparison result = "+
                           i1.compareTo(i1));   // 0
        // The following line throws an exception:
        // System.out.println("The comparison result = "+
        //                    i1.compareTo(i3));

    }
}
/*
Output:
The comparison result = 1
The comparison result = -1
The comparison result = 0
*/
```

Remarks

If the parameter objects are an Integer, the program will throw a java.lang.ClassCastException exception.

See Also

Reference

Integer Class
Concepts
Integer Members
java.lang Package

Integer.decode Method

Decodes a string and returns the corresponding integer.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Integer decode(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be decoded.

Return Value

The Integer object that corresponds to the string parameter.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.doubleValue Method

Returns the double value of an Integer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double doubleValue();
```

Return Value

The double value of the Integer object.

Example

The following example uses the method doubleValue to get the equivalent double number of an Integer object.

```
// i-doubleVal1.jsl
// compareTo Example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer i = new Integer(123);

        System.out.println("The double value is: " + i.doubleValue());
   }
}
/*
Output:
The double value is: 123.0
*/
```

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.equals Method

Compares the current object to another object and returns a Boolean value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

true if the int values of the two objects are identical, false otherwise.

Example

```
// i-equals.jsl
// equals Example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer i1 = new Integer(123);
        Integer i2 = new Integer (123);

        System.out.print("The equality is: "+ i1.equals(i2));
    }
}

/*
Output:
The equality is: true
*/
```

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.floatValue Method

Returns the float value of an Integer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float floatValue();
```

Return Value

The float value of the Integer object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.hashCode Method

Retrieves the hash code of an Integer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

T the hash code of an Integer object, which is the value of the primitive int stored in the object.

Example

```
// i-GHashCode1.jsl
// compareTo Example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer i = new Integer(423);

        System.out.println("The hash code is: " + i.GetHashCode());
   }
}
/*
Output:
The hash code is: 423
*/
```

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.getInteger Method

Returns the integer value of a property.

Overload List

Name	Description
Integer.getInteger (String)	Returns the integer value of a property with a specific name.
Integer.getInteger (String, int)	Returns the integer value of a property with a specific name and a default value.
Integer.getInteger (String, Integer)	Returns the integer value of a property with a specific name and a default value.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.getInteger Method (String)

Returns the integer value of a property with a specific name.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Integer getInteger(
    java.lang.String prop);
```

Parameters

prop

The property name.

Return Value

The integer value of the property.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.getInteger Method (String, Int32)

Returns the integer value of a property with a specific name and a default value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Integer getInteger(
    java.lang.String prop,
    int defval);
```

Parameters

prop

The property name.

defval

The property default value.

Return Value

The integer value of the property.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.getInteger Method (String, Integer)

Returns the integer value of a property with a specific name and a default value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Integer getInteger(
    java.lang.String prop,
    java.lang.Integer defobj);
```

Parameters

prop

The property name.

defobj

The property default value.

Return Value

The integer value of the property.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.intValue Method

Returns the int value of the current Integer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int intValue();
```

Return Value

The primitive value stored in the Integer object.

Example

```
// IntegerValue.jsl
// intValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer i = new Integer(123);

        System.out.print("The integer value is: "+ i.intValue());
    }
}

/*
Output:
The integer value is: 123
*/
```

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.longValue Method

Returns the long value of the current Integer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long longValue();
```

Return Value

The long value of the Integer object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.parseInt Method

Parses the string parameter and converts it to an integer value.

Overload List

Name	Description
Integer.parseInt (String)	Parses the string parameter and converts it to an integer value.
Integer.parseInt (String, int)	Parses the string parameter and converts it to an integer value using a specified radix.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.parseInt Method (String)

Parses the string parameter and converts it to an integer value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int parseInt(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string parameter.

Return Value

The integer value of the parsed string.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.parseInt Method (String, Int32)

Parses the string parameter and converts it to an integer value using a specified radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int parseInt(
    java.lang.String str,
    int radix) throws java.lang.NumberFormatException;
```

Parameters

str

The string parameter.

radix

The specified radix.

Return Value

The integer value of the parsed string in the specified radix.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.shortValue Method

Returns the short value of an Integer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short shortValue();
```

Return Value

The short value of the Integer object.

Example

```
// i-shortValue1.jsl
// shortValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Integer d = new Integer(123);

        System.out.println("The short value is: "+ d.shortValue());
    }
}

/*
Output:
The short value is: 123
*/
```

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.toBinaryString Method

Converts the value of an integer to a binary string.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toBinaryString(
    int inum);
```

Parameters

inum

An int expression.

Return Value

The binary string representation of the integer parameter.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToBoolean Method

Converts the value of an integer to Boolean using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean ToBoolean(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

true if successful; false otherwise.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToByte Method

Converts an Integer value to a byte using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Byte ToByte(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The byte value of the object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToChar Method

Converts an Integer value to a char using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public char ToChar(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The char value of the Integer object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToDateTime Method

Converts an Integer object value to a date-time value using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.DateTime ToDateTime(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The date-time value of the object using the specific format.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToDecimal Method

Converts an Integer object value to a decimal value using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Decimal ToDecimal(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The decimal value of the object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToDouble Method

Converts an Integer object value to a double value using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double ToDouble(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The double value of the object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.toHexString Method

Returns the hexadecimal string representation of an int.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toHexString(
    int inum);
```

Parameters

inum

The int value to be represented as hexadecimal.

Return Value

The string representation of the parameter.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToInt16 Method

Converts an Integer object value to a 2-byte integer using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short ToInt16(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The 2-byte integer value of the object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToInt32 Method

Converts an Integer object value to a 4-byte integer using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int ToInt32(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The 4-byte integer value of the object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToInt64 Method

Converts an Integer object value to an 8-byte integer using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long ToInt64(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The 8-byte integer value of the object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.toOctalString Method

Returns the octal number string representation of an int.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toOctalString(
    int inum);
```

Parameters

inum

The int parameter whose octal representation is required.

Return Value

The octal number representation of the parameter.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToSByte Method

Converts an Integer object value to an sbyte value using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte ToSByte(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The sbyte value of the object.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToSingle Method

Converts an Integer object value to a float value using a specific format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float ToSingle(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The float value of the parameter.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToString Method

Returns the string representation of the value of an Integer object.

Overload List

Name	Description
Integer.ToString ()	Returns the string representation of the value of an Integer object.
Integer.ToString (IFormatProvider)	Returns the string representation of the value of an Integer object using a specified format.
Integer.toString (Int32)	Converts an int to its string representation.
Integer.toString (Int32, Int32)	Returns the string representation of the specified int parameter using the number system of the specified radix.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToString Method ()

Returns the string representation of the value of an Integer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The string representation of the value of the Integer object.

Example

The following example converts the value of the constant Integer.MAX_VALUE to a string and displays the result.

```
// i-toString.jsl
// Integer toString example

public class MyClass
{
   public static void main(String[] args)
   {
      Integer i = new Integer(Integer.MAX_VALUE);

      System.out.println("The string is: " + i.toString());
      // or:
      // System.out.println("The string is: " + i);
   }
}

/*
Output:
The string is: 2147483647
*/
```

Remarks

The method is implicitly included in the print and println methods.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToString Method (IFormatProvider)

Returns the string representation of the value of an Integer object using a specified format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String ToString(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The string representation of the value of the object using the specified format.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.toString Method (Int32)

Converts an int to its string representation.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toString(
    int inum);
```

Parameters

inum

Expression of the type int.

Return Value

The string representation of the specified int parameter.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.toString Method (Int32, Int32)

Returns the string representation of the specified int parameter using the number system of the specified radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toString(
    int inum,
    int radix);
```

Parameters

inum

The int to be converted.

radix

The radix of the number system.

Return Value

The string representation of the specified int parameter using the number system of the specified radix.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToType Method

Converts an Integer object to a specific type using a specified format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object ToType(
    System.Type conversionType,
    System.IFormatProvider provider);
```

Parameters

conversionType

The type to convert to.

provider

The format provider.

Return Value

The object converted to the specified type in the specified format.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToUInt16 Method

Converts an Integer object to an unsigned short integer using a specified format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt16 ToUInt16(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The unsigned short integer.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToUInt32 Method

Converts an Integer object to an unsigned integer using a specified format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt32 ToUInt32(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The unsigned integer in the specified format.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.ToUInt64 Method

Converts an Integer object to an unsigned long integer using a specified format.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt64 ToUInt64(
    System.IFormatProvider provider);
```

Parameters

provider

The format provider.

Return Value

The unsigned long integer in the specified format.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.valueOf Method

Overload List

Name	Description
Integer.valueOf (String)	Returns the value of an Integer object represented by a specified string parameter.
Integer.valueOf (String, int)	Returns the integer value represented by a specified string parameter in the radix specified by an integer parameter.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.valueOf Method (String)

Returns the value of an Integer object represented by a specified string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Integer valueOf(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string parameter.

Return Value

The integer value represented by the string parameter.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

Integer.valueOf Method (String, Int32)

Returns the integer value represented by a specified string parameter in the radix specified by an integer parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Integer valueOf(
    java.lang.String str,
    int radix) throws java.lang.NumberFormatException;
```

Parameters

str

The string representing the number.

radix

The integer representing the radix.

Return Value

The integer value represented by the string in the specified radix.

See Also

Reference

Integer Class

Concepts

Integer Members

java.lang Package

InternalError Class

An error thrown to indicate that an internal error has occurred.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.InternalError
    extends java.lang.VirtualMachineError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.VirtualMachineError

java.lang.InternalError

See Also

Concepts

InternalError Members

java.lang Package

InternalError Members

An error thrown to indicate that an internal error has occurred.

The following tables list the members exposed by the InternalError type.

Public Constructors

Name	Description
InternalError	Overloaded. Constructs an InternalError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an error object to an InternalError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InternalError Class

Concepts

java.lang Package

InternalError Constructor

Constructs an InternalError object.

Overload List

Name	Description
InternalError ()	Constructs an InternalError object with no specific details.
InternalError (String)	Constructs an InternalError object with specific text message.
InternalError (SerializationInfo, StreamingContext)	Constructs an InternalError object during serialization.
InternalError (String, Exception)	Constructs an InternalError object with a text message and inner exception.

See Also

Reference

InternalError Class

Concepts

InternalError Members

java.lang Package

InternalError Constructor ()

Constructs an InternalError object with no specific details.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InternalError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

InternalError Class

Concepts

InternalError Members

java.lang Package

InternalError Constructor (String)

Constructs an InternalError object with specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InternalError(
    java.lang.String s);
```

Parameters

s

The string that contains the error message text.

See Also

Reference

InternalError Class

Concepts

InternalError Members

java.lang Package

InternalError Constructor (SerializationInfo, StreamingContext)

Constructs an InternalError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.InternalError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

InternalError Class

Concepts

InternalError Members

java.lang Package

InternalError Constructor (String, Exception)

Constructs an InternalError object with a text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InternalError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to InternalError.

See Also

Reference

InternalError Class

Concepts

InternalError Members

java.lang Package

InternalError Methods

Public Methods

Name	Description
<exceptFilter>	Maps an error object to an InternalError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InternalError Class

Concepts

java.lang Package

InternalError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

InternalError Class

Concepts

java.lang Package

InterruptedException Class

The exception that is thrown when a thread is interrupted.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.InterruptedException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.InterruptedException

See Also

Concepts

InterruptedException Members

java.lang Package

InterruptedException Members

The exception that is thrown when a thread is interrupted.

The following tables list the members exposed by the InterruptedException type.

Public Constructors

Name	Description
InterruptedException	Overloaded. Initializes a new instance of an InterruptedException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InterruptedException Class

Concepts

java.lang Package

InterruptedException Constructor

Initializes a new instance of an InterruptedException object.

Overload List

Name	Description
InterruptedException ()	Initializes a new instance of an InterruptedException object.
InterruptedException (String)	Initializes a new instance of an InterruptedException object with the given message.
InterruptedException (SerializationInfo, StreamingContext)	Initializes a new instance of an InterruptedException object during deserialization.
InterruptedException (String, Exception)	Initializes a new instance of an InterruptedException object with the given message and inner exception.

See Also

Reference

InterruptedException Class

Concepts

InterruptedException Members

java.lang Package

InterruptedException Constructor ()

Initializes a new instance of an InterruptedException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InterruptedException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

InterruptedException Class

Concepts

InterruptedException Members

java.lang Package

InterruptedException Constructor (String)

Initializes a new instance of an InterruptedException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InterruptedException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also

Reference

InterruptedException Class

Concepts

InterruptedException Members

java.lang Package

InterruptedException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an InterruptedException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.InterruptedException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

InterruptedException Class

Concepts

InterruptedException Members

java.lang Package

InterruptedException Constructor (String, Exception)

Initializes a new instance of an InterruptedException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.InterruptedException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the InterruptedException being thrown.

See Also

Reference

InterruptedException Class

Concepts

InterruptedException Members

java.lang Package

InterruptedException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InterruptedException Class

Concepts

java.lang Package

InterruptedException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

InterruptedException Class

Concepts

java.lang Package

LinkageError Class

The error that is thrown when there is an error linking an application.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.LinkageError
    extends java.lang.Error
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

Derived Classes

See Also

Concepts

LinkageError Members

java.lang Package

LinkageError Members

The error that is thrown when there is an error linking an application.

The following tables list the members exposed by the LinkageError type.

Public Constructors

Name	Description
LinkageError	Overloaded. Initializes a new instance of a LinkageError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Do not document.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

LinkageError Class

Concepts

java.lang Package

LinkageError Constructor

Initializes a new instance of a LinkageError object.

Overload List

Name	Description
LinkageError ()	Initializes a new instance of a LinkageError object.
LinkageError (String)	Initializes a new instance of a LinkageError object with the given message.
LinkageError (SerializationInfo, StreamingContext)	Initializes a new instance of a LinkageError object during deserialization.
LinkageError (String, Exception)	Initializes a new instance of a LinkageError object with the given message and inner exception.

See Also

Reference

LinkageError Class

Concepts

LinkageError Members

java.lang Package

LinkageError Constructor ()

Initializes a new instance of a LinkageError object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.LinkageError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

LinkageError Class

Concepts

LinkageError Members

java.lang Package

LinkageError Constructor (String)

Initializes a new instance of a LinkageError object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.LinkageError(
    java.lang.String s);
```

Parameters

s

A message describing the error condition.

See Also

Reference

LinkageError Class

Concepts

LinkageError Members

java.lang Package

LinkageError Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a LinkageError object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.LinkageError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

LinkageError Class

Concepts

LinkageError Members

java.lang Package

LinkageError Constructor (String, Exception)

Initializes a new instance of a LinkageError object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.LinkageError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the error condition.

inner

An inner exception that led to the LinkageError being thrown.

See Also

Reference

LinkageError Class

Concepts

LinkageError Members

java.lang Package

LinkageError Methods

Public Methods

Name	Description
<exceptFilter>	Do not document.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

LinkageError Class

Concepts

java.lang Package

LinkageError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

LinkageError Class

Concepts

java.lang Package

Long Class

Wraps the primitive type long into an object. It also contains methods for conversion and processing Long objects.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Long
    extends java.lang.Number
    implements System.IConvertible, java.lang.Comparable
```

Remarks

This class implements System.IConvertible.

Inheritance Hierarchy

java.lang.Object

  java.lang.Number

   java.lang.Long

See Also

Concepts

Long Members

java.lang Package

Long Members

Wraps the primitive type long into an object. It also contains methods for conversion and processing Long objects.

The following tables list the members exposed by the Long type.

Public Constructors

Name	Description
Long	Overloaded.

Public Fields

Name	Description
MAX_VALUE	A constant that stores the maximum value a long number can have.
MIN_VALUE	A constant that stores the minimum value a long number can have.
TYPE	A readonly value that represents the Class instance of the primitive type long.

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value stored in a Long object.
compareTo	Overloaded.
decode	Decodes a string parameter to a Long object.
doubleValue	Overridden. Returns the value of a Long object converted to double.
equals	Overridden. Compares the current Long object to another object.
floatValue	Overridden. Returns value of a Long object converted to float.
hashCode	Overridden. Returns the hash code of a Long object.
getLong	Overloaded.
getClass	(inherited from Object)
intValue	Overridden. Returns the value of a Long object as an int.
longValue	Overridden. Returns the value of a Long object as a long number.
clone	(inherited from Number)
parseLong	Overloaded.
shortValue	Overridden. Returns the value of a Long object as a short number.
toBinaryString	Returns the binary string that represents a long value.
ToBoolean	

ToByte	
ToChar	
ToDateTime	
ToDecimal	
ToDouble	
toHexString	Returns the hexadecimal string that represents a long value.
ToInt16	
ToInt32	
ToInt64	
toOctalString	Returns the octal string that represents the value of a long value.
ToSByte	
ToSingle	
toString	Overloaded.
toString	Overloaded. Overridden.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Overloaded.

See Also

Reference

Long Class

Concepts

java.lang Package

Long Fields

Public Fields

Name	Description
MAX_VALUE	A constant that stores the maximum value a long number can have.
MIN_VALUE	A constant that stores the minimum value a long number can have.
TYPE	A readonly value that represents the Class instance of the primitive type long.

See Also

Reference

Long Class

Concepts

java.lang Package

Long.MAX_VALUE Field

A constant that stores the maximum value a long number can have.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final long MAX_VALUE;
```

Example

```
// l-MAX_VALUE.jsl
// Long MAX_VALUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("MAX_VALUE = " + Long.MAX_VALUE);
    }
}

/*
Output:
MAX_VALUE = 9223372036854775807
*/
```

Remarks

The value of MAX_VALUE is 9223372036854775807.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.MIN_VALUE Field

A constant that stores the minimum value a long number can have.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final long MIN_VALUE;
```

Example

```
// l-MIN_VALUE.jsl
// Long MIN_VALUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("MIN_VALUE = " + Long.MIN_VALUE);
    }
}

/*
Output:
MIN_VALUE = -9223372036854775808
*/
```

Remarks

The value of MIN_VALUE is -9223372036854775808.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.TYPE Field

A readonly value that represents the Class instance of the primitive type long.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Class TYPE;
```

Example

```
// l-TYPE.jsl
// Long TYPE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("TYPE = " + Long.TYPE);
    }
}

/*
Output:
TYPE = Long
*/
```

Remarks

The value of TYPE is long.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long Constructor

Overload List

Name	Description
Long (long)	Constructs a new Long object that represents a specified long value.
Long (String)	Constructs a new Long object that represents the long value specified by a string parameter.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long Constructor (Int64)

Constructs a new Long object that represents a specified long value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Long(
    long value);
```

Parameters

value

A long expression.

Example

```
// l-ctor1.jsl
// Long.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Long l = new Long(123L);
        System.out.print("The long value is: " + l);
    }
}

/*
Output:
The long value is: 123
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long Constructor (String)

Constructs a new Long object that represents the long value specified by a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Long(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

A string expression.

Example

```
// l-ctor2.jsl
// Long.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Long l = new Long("123");
        System.out.print("The long value is: " + l);
    }
}


/*
Output:
The long value is: 123
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long Methods

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value stored in a Long object.
compareTo	Overloaded.
decode	Decodes a string parameter to a Long object.
doubleValue	Overridden. Returns the value of a Long object converted to double.
equals	Overridden. Compares the current Long object to another object.
floatValue	Overridden. Returns value of a Long object converted to float.
hashCode	Overridden. Returns the hash code of a Long object.
getLong	Overloaded.
getClass	(inherited from Object)
intValue	Overridden. Returns the value of a Long object as an int.
longValue	Overridden. Returns the value of a Long object as a long number.
clone	(inherited from Number)
parseLong	Overloaded.
shortValue	Overridden. Returns the value of a Long object as a short number.
toBinaryString	Returns the binary string that represents a long value.
ToBoolean	
ToByte	
ToChar	
ToDateTime	
ToDecimal	
ToDouble	
toHexString	Returns the hexadecimal string that represents a long value.
ToInt16	
ToInt32	

ToInt64	
toOctalString	Returns the octal string that represents the value of a long value.
ToSByte	
ToSingle	
toString	Overloaded.
toString	Overloaded. Overridden.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Overloaded.

See Also

Reference

Long Class

Concepts

java.lang Package

Long.byteValue Method

Returns the byte value stored in a Long object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte byteValue();
```

Return Value

The byte value stored in a Long object.

Example

```
// l-byteval1.jsl
// Long.byteValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Long l = new Long("123");
        System.out.print("The byte value is: " + l.byteValue());
    }
}

/*
Output:
The byte value is: 123
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.compareTo Method

Overload List

Name	Description
Long.compareTo (Long)	Compares the current Long object to another Long object.
Long.compareTo (Object)	Compares the current object to another object.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.compareTo Method (Long)

Compares the current Long object to another Long object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Long aLong);
```

Parameters

aLong

The Long object to compare to.

Return Value

0 if the two objects are identical. Negative value if aLong is greater than the current object. Positive value if aLong is less than the current object.

Example

```
// l-comp1.jsl
// Long.compareTo example

public class MyClass
{
    public static void main(String[] args)
    {
        Long l1 = new Long("121");
        Long l2 = new Long("123");
        System.out.print("The result is: " + l1.compareTo(l2));
    }
}

/*
Output:
The result is: -1
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.compareTo Method (Object)

Compares the current object to another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

If obj is a Long object, it behaves exactly like compareTo, in which case it returns:

0 if the two objects are identical.

Negative value if obj is greater than the current object.

Positive value if obj is less than the current object.

If obj is of another type other than Long, the program throws the exception java.lang.ClassCastException.

Example

```
// Long.compareTo example
// Throws an exception.

public class MyClass
{
    public static void main(String[] args)
    {
        Long l = new Long("121");
        Integer i = new Integer("123");
        System.out.print("The result is: " + l.compareTo(i));
    }
}

/*
Output:
java.lang.ClassCastException: Specified cast is not valid.
   at java.lang.Long.compareTo(Object obj)
   at MyClass.main(String[] args)
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.decode Method

Decodes a string parameter to a Long object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Long decode(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be decoded.

Return Value

The Long object that contains the long value of str.

Example

```
// l-decode1.jsl
// Long.decode example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("0x27");   // Hexadecimal number
        Long l = Long.decode(s);
        System.out.print("The decoded Long is: " + l);
    }
}

/*
Output:
The decoded long is: 39
*/
```

Remarks

The string parameter str might contain decimal, hexadecimal, or octal digits.

Hexadecimal digits are preceded by "0x", 0X", or "#".

Octal digits are preceded by zero (0).

If the string cannot be parsed to a long value, the exception java.lang.NumberFormatException is thrown.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.doubleValue Method

Returns the value of a Long object converted to double.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double doubleValue();
```

Return Value

The value of a Long object converted to double.

Example

```
// l-doubleValue1.jsl
// Long.doubleValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Long l = new Long("123");
        // Convert it to double and add it to another double number:
        double d = l.floatValue() + 321.4;
        System.out.print("The new double number is: " + d);
    }
}

/*
Output:
The new double number is: 444.4
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.equals Method

Compares the current Long object to another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

true if the two objects are identical; false otherwise.

Example

```
// l-equals1.jsl
// Long.Equals example

public class MyClass
{
    public static void main(String[] args)
    {
        Long l1 = new Long("123");
        Long l2 = new Long("123");
        System.out.print("The result is: " + l1.Equals(l2));
    }
}

/*
The result is: true
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.floatValue Method

Returns value of a Long object converted to float.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float floatValue();
```

Return Value

The value of the Long object converted to float.

Example

```
// l-floatValue1.jsl
// Long.floatValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Long l = new Long("123");
        // Convert it to float and add it to another float number:
        float f = l.floatValue() + 321.4F;
        System.out.print("The new float number is: " + f);
    }
}

/*
Output:
The new float numler is: 444.4
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.hashCode Method

Returns the hash code of a Long object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code of the Long object.

Example

```
// l-getHash1.jsl
// Long.GetHashCode example

public class MyClass
{
    public static void main(String[] args)
    {
        Long b = new Long("123");
        int h = b.GetHashCode();
        System.out.print("The hash code is: " + h);
    }
}

/*
The hash code is: 123
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.getLong Method

Overload List

Name	Description
Long.getLong (String)	Returns the long value of the system property specified by a string.
Long.getLong (String, long)	Returns a long value that represents the system property with the specified name.
Long.getLong (String, Long)	Returns a long value that represents the system property with the specified name.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.getLong Method (String)

Returns the long value of the system property specified by a string.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Long getLong(
    java.lang.String prop);
```

Parameters

prop

The system property name.

Return Value

The long value of prop.

Example

```
// l-getLong1.jsl
// Long.getLong example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = System.getProperty("os.version");
        // Notice that the output from the following statement
        // can be different on different machines:
        System.out.println("os.version: " + s);
        System.out.println("The long value is: " + Long.getLong(s));
    }
}

/*
Output:
os.version: 5.1
The long value is: null
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.getLong Method (String, Int64)

Returns a long value that represents the system property with the specified name.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Long getLong(
    java.lang.String prop,
    long defval);
```

Parameters

prop

The property name.

defval

The long default value.

Return Value

The long value that represents the system property prop.

Example

```
// l-getLong2.jsl
// Long.getLong example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = System.getProperty("os.version");
        // Notice that the output from the following statement
        // can be different on different machines:
        System.out.println("os.version: " + s);
        System.out.println("The value is: " +
                           Long.getLong(s,123));
    }
}

/*
os.version: 5.1
The value is: 123
*/
```

Remarks

If the value that represents the system property is null, the default value defval is returned.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.getLong Method (String, Long)

Returns a long value that represents the system property with the specified name.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Long getLong(
    java.lang.String prop,
    java.lang.Long defobj);
```

Parameters

prop

The property name.

defobj

The default value object.

Return Value

The long value that represents the system property prop.

Example

```
// l-getLong3.jsl
// Long.getLong example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = System.getProperty("os.version");
        // The default object:
        Long l = new Long("123");

        // Notice that the output from the following statement
        // can be different on different machines:
        System.out.println("os.version: " + s);
        System.out.println("The value is: " + Long.getLong(s,l));
    }
}

/*
os.version: 5.1
The value is: 123
*/
```

Remarks

If the value that represents the system property is null, defobj is returned.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.intValue Method

Returns the value of a Long object as an int.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int intValue();
```

Return Value

The value of the Long object as an int.

Example

```
// l-intValue1.jsl
// Long.intValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Long b = new Long("123");
        // Convert it and add it to another int:
        int i = b.intValue() + 321;
        System.out.print("The new int is: " + i);
    }
}

/*
Output:
The new int is: 444
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.longValue Method

Returns the value of a Long object as a long number.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long longValue();
```

Return Value

The value of the Long object as a long number.

Example

```
// l-longValue1.jsl
// Long.longValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Long b = new Long("123");
        // Convert it and add it to another long number:
        long l = b.longValue() + 321L;
        System.out.print("The new long number is: " + l);
    }
}

/*
Output:
The new long number is: 444
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.parseLong Method

Overload List

Name	Description
Long.parseLong (String)	Returns the long value represented by a string parameter.
Long.parseLong (String, int)	Returns the signed long value represented by a string parameter in the specified radix.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.parseLong Method (String)

Returns the long value represented by a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static long parseLong(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

Return Value

The long value represented by the string parameter.

Example

```
// l-parseLong1.jsl
// Long.parseLong example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("123");
        long l = Long.parseLong(s);
        System.out.print("The long value is: " + l);
    }
}

/*
Output:
The long value is: 123
*/
```

Remarks

The string characters must all be decimal digits except the first character, which can be a minus sign (-).

The returned value is a signed long in the decimal radix.

If the string could not be parsed to a signed decimal long, the exception java.lang.NumberFormatException is thrown.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.parseLong Method (String, Int32)

Returns the signed long value represented by a string parameter in the specified radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static long parseLong(
    java.lang.String str,
    int radix) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

radix

The radix used in parsing str.

Return Value

The long value represented by str in the specified radix.

Example

```
// l-parse2.jsl
// Long.parseLong example

public class MyClass
{
    public static void main(String[] args)
    {
        String s1 = new String("255");   // Decimal
        String s2 = new String("FF");    // Hexadecimal

        long l1 = Long.parseLong(s1,10);
        long l2 = Long.parseLong(s2,16);
        System.out.print("l1 = " + l1 + "\nl2 = " + l2);
    }
}

/*
Output:
l1 = 255
l2 = 255
*/
```

Remarks

The string characters must all be digits in the specified radix except the first character, which can be a minus sign (-).

The exception java.lang.NumberFormatException is generated if any of the following conditions occurred:

The string parameter is null.

The length of the string parameter is zero ("").

If the value of the radix is not in the range Character.MIN_RADIX to Character.MAX_RADIX.

The string does not represent a long value.

If any of the characters in the string, except the first character, which can be a minus sign, is not a valid digit in the specified radix.

See Also

Reference
Long Class
Concepts
Long Members
java.lang Package

Long.shortValue Method

Returns the value of a Long object as a short number.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short shortValue();
```

Return Value

The value of the Long object as a short number.

Example

```
// l-shvalue1.jsl
// Long.shortValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Long l = new Long("123");
        short s = l.shortValue();
        System.out.print("The short value is: " + s);
    }
}

/*
Output:
The short value is: 123
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.toBinaryString Method

Returns the binary string that represents a long value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toBinaryString(
    long lnum);
```

Parameters

lnum

A long expression.

Return Value

The binary string that represents the long value.

Example

```
// l-toBStr1.jsl
// Long.toBinaryString example

public class MyClass
{
    public static void main(String[] args)
    {
        long l = 124;
        // Converting to a string and catenating to another string:
        String s = "The string is: " + Long.toBinaryString(l);
        System.out.print(s);
    }
}

/*
The string is: 1111100
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToBoolean Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean ToBoolean(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Byte ToByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToChar Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public char ToChar(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToDateTime Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.DateTime ToDateTime(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToDecimal Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Decimal ToDecimal(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToDouble Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double ToDouble(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.toHexString Method

Returns the hexadecimal string that represents a long value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toHexString(
    long lnum);
```

Parameters

lnum

A long expression.

Return Value

The hexadecimal string that represents the long value.

Example

```
// l-toHStr1.jsl
// Long.toHexString example

public class MyClass
{
    public static void main(String[] args)
    {
        long l = 255;
        // Converting to a string and catenating to another string:
        String s = "The string is: " + Long.toHexString(l);
        System.out.print(s);
    }
}

/*
The string is: ff
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short ToInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int ToInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long ToInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.toOctalString Method

Returns the octal string that represents the value of a long value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toOctalString(
    long lnum);
```

Parameters

lnum

A long expression.

Return Value

The octal string that represents the value of a long value.

Example

```
// l-toOStr1.jsl
// Long.toOctalString example

public class MyClass
{
    public static void main(String[] args)
    {
        long l = 254;
        // Converting to a string and catenating to another string:
        String s = "The string is: " + Long.toOctalString(l);
        System.out.print(s);
    }
}

/*
The string is: 376
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToSByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte ToSByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToSingle Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float ToSingle(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToString Method

Overload List

Name	Description
Long.ToString ()	Returns the string that represents the value of a Long object.
Long.ToString (IFormatProvider)	
Long.toString (Int64)	Returns the string that represents a long value.
Long.toString (Int64, Int32)	Returns the string that represents a long value in the specified radix.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToString Method ()

Returns the string that represents the value of a Long object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The string that represents the value of the current Long object

Example

```
// l-toStr3.jsl
// Long.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        Long l = new Long(255);
        // Converting to a string and catenating to another string:
        String s = "The string is: " + l.toString();
        System.out.print(s);
    }
}

/*
The string is: 255
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToString Method (IFormatProvider)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String ToString(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.toString Method (Int64)

Returns the string that represents a long value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toString(
    long lnum);
```

Parameters

lnum

A long expression.

Return Value

The string that represents the long value.

Example

```
// l-toStr1.jsl
// Long.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        long l = 254;
        // Converting to a string and catenating to another string:
        String s = "The string is: " + Long.toString(l);
        System.out.print(s);
    }
}

/*
The string is: 254
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.toString Method (Int64, Int32)

Returns the string that represents a long value in the specified radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toString(
    long lnum,
    int radix);
```

Parameters

lnum

A long expression.

radix

The specified radix.

Return Value

The string that represents the long value in the specified radix.

Example

```
// l-toStr2.jsl
// Long.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        long l = 255;
        // Converting to a string and catenating to another string,
        // using the hexadecimal radix:
        String s = "The string is: " + Long.toString(l,16);
        System.out.print(s);
    }
}

/*
The string is: ff
*/
```

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToType Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object ToType(
    System.Type conversionType,
    System.IFormatProvider provider);
```

Parameters

conversionType

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToUInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt16 ToUInt16(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToUInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt32 ToUInt32(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.ToUInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt64 ToUInt64(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.valueOf Method

Overload List

Name	Description
Long.valueOf (String)	Returns the Long object represented by a string parameter.
Long.valueOf (String, int)	Returns the Long object represented by a string parameter in the specified radix.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.valueOf Method (String)

Returns the Long object represented by a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Long valueOf(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

Return Value

The Long object represented by str.

Example

```
// l-ValueOf1.jsl
// Long.valueOf example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("-128");
        Long l = Long.valueOf(s);
        System.out.print("The Long is: " + l);
    }
}

/*
Output:
The Long is: -128
*/
```

Remarks

The string characters must all be decimal digits except the first character, which can be a minus sign (-).

If the string could not be parsed to a signed decimal long, the exception java.lang.NumberFormatException is thrown.

See Also

Reference

Long Class

Concepts

Long Members

java.lang Package

Long.valueOf Method (String, Int32)

Returns the Long object represented by a string parameter in the specified radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Long valueOf(
    java.lang.String str,
    int radix) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

radix

The radix used in parsing str.

Return Value

The Long object represented by str in the specified radix.

Example

```
// l-ValueOf2.jsl
// Long.valueOf example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("-80");
        // Parse to a hexadecimal number:
        Long l = Long.valueOf(s,16);
        System.out.print("The long value is: " + l);
    }
}

/*
Output:
The long value is: -128
*/
```

Remarks

The string characters must all be digits in the specified radix except the first character, which can be a minus sign (-).

The exception java.lang.NumberFormatException is generated if any of the following conditions occurred:

The string parameter is null.

The length of the string parameter is zero ("").

If the value of the radix is not in the range Character.MIN_RADIX to Character.MAX_RADIX.

The string does not represent a long value.

If any of the characters in the string, except the first character, which can be a minus sign, is not a valid digit in the specified radix.

See Also

Reference

Long Class

Concepts

Long Members
java.lang Package

Math Class

Declares the Math class type, which contains method members to perform basic numeric operations such as exponentiation, logarithms, and trigonometric functions.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Math
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.lang.Math

See Also

Concepts

Math Members

java.lang Package

Math Members

Declares the Math class type, which contains method members to perform basic numeric operations such as exponentiation, logarithms, and trigonometric functions.

The following tables list the members exposed by the Math type.

Public Constructors

Name	Description
Math	Constructs a new Math object.

Public Fields

Name	Description
E	Returns the value of the base of the natural system of logarithms (e).
PI	Returns the approximate value of the ratio of a circle's circumference to its diameter (PI).

Public Methods

Name	Description
abs	Overloaded.
acos	Returns the arc cosine of a double value.
asin	Returns the arc sine of a double value.
atan	Returns the arc tangent of double value.
atan2	Converts the Cartesian coordinates of a point (x,y) to the polar coordinates by computing the arc tangent for y/x.
ceil	Returns a double number that represents the smallest mathematical integer that is not less than a double parameter.
cos	Returns the cosine of double parameter.
equals	(inherited from Object)
exp	Returns the value of the base of the natural system of logarithms (e) raised to the power of a double parameter.
floor	Returns a double number that represents the largest mathematical integer that is not greater than a double parameter.
hashCode	(inherited from Object)
getClass	(inherited from Object)
IEEEremainder	Returns the remainder of the integer division of a double parameter by a second double parameter.
log	Returns the natural logarithm of a double parameter.
max	Overloaded.

clone	
min	Overloaded.
pow	Returns the value of a base double number raised to the power of a double exponent.
random	Returns a random double value between 0.0 and 1.0.
rint	Returns a double number that represents the closest integer to a double parameter.
round	Overloaded.
sin	Returns the sine of a double angle.
sqrt	Returns the square root of a double parameter.
tan	Returns the tangent of a double angle.
toDegrees	Converts an angle from radians angle to degrees.
toRadians	Converts an angle from degrees to radians.
toString	Overridden. Returns the string representation of an object.

See Also

Reference

Math Class

Concepts

java.lang Package

Math Fields

Public Fields

Name	Description
E	Returns the value of the base of the natural system of logarithms (e).
PI	Returns the approximate value of the ratio of a circle's circumference to its diameter (PI).

See Also

Reference

Math Class

Concepts

java.lang Package

Math.E Field

Returns the value of the base of the natural system of logarithms (e).

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final double E;
```

Example

This example displays the value of the field E.

```
// m-E.jsl
// Math.E example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("Euler's number = " + Math.E);
    }
}

/*
Output:
Euler's number = 2.7182818284590451
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.PI Field

Returns the approximate value of the ratio of a circle's circumference to its diameter (PI).

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final double PI;
```

Example

This example displays the value of the field PI.

```
// m-PI.jsl
// Math.PI example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The approximate value of PI = " + Math.PI);
    }
}

/*
Output:
The approximate value of PI = 3.1415926535897931
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math Constructor

Constructs a new Math object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Math();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math Methods

Public Methods

Name	Description
abs	Overloaded.
acos	Returns the arc cosine of a double value.
asin	Returns the arc sine of a double value.
atan	Returns the arc tangent of double value.
atan2	Converts the Cartesian coordinates of a point (x,y) to the polar coordinates by computing the arc tangent for y/x.
ceil	Returns a double number that represents the smallest mathematical integer that is not less than a double parameter.
cos	Returns the cosine of double parameter.
equals	(inherited from Object)
exp	Returns the value of the base of the natural system of logarithms (e) raised to the power of a double parameter.
floor	Returns a double number that represents the largest mathematical integer that is not greater than a double parameter.
hashCode	(inherited from Object)
getClass	(inherited from Object)
IEEEremainder	Returns the remainder of the integer division of a double parameter by a second double parameter.
log	Returns the natural logarithm of a double parameter.
max	Overloaded.
clone	
min	Overloaded.
pow	Returns the value of a base double number raised to the power of a double exponent.
random	Returns a random double value between 0.0 and 1.0.
rint	Returns a double number that represents the closest integer to a double parameter.
round	Overloaded.
sin	Returns the sine of a double angle.
sqrt	Returns the square root of a double parameter.

tan	Returns the tangent of a double angle.
toDegrees	Converts an angle from radians angle to degrees.
toRadians	Converts an angle from degrees to radians.
toString	Overridden. Returns the string representation of an object.

See Also

Reference

Math Class

Concepts

java.lang Package

Math.abs Method

Overload List

Name	Description
Math.abs (double)	Returns the absolute value of a double parameter.
Math.abs (int)	Returns the absolute value of an int parameter.
Math.abs (long)	Returns the absolute value of long parameter.
Math.abs (float)	Returns the absolute value of float parameter.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.abs Method (Double)

Returns the absolute value of a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double abs(
    double a);
```

Parameters

a

A double expression.

Return Value

The absolute value of the parameter.

Example

The following example displays some double values and their absolute values.

```
// m-absDouble.jsl
// Math.abs example using Double

public class MyClass
{
    public static void main(String[] args)
    {
        double x = Double.MAX_VALUE;
        double y = Double.MIN_VALUE;
        double z = -88.1;

// Original values:
        System.out.println("The original values:");
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);

// Absolute values:
        System.out.println("The absolute values:");
        System.out.println(Math.abs(x));
        System.out.println(Math.abs(y));
        System.out.println(Math.abs(z));
    }
}

/*
Output:
The original values:
1.7976931348623157E308
4.94065645841247E-324
-88.1
The absolute values:
1.7976931348623157E308
4.94065645841247E-324
88.1
*/
```

Remarks

If the parameter is not negative, the parameter is returned.

If the parameter is negative, the negation of the parameter is returned.

If the parameter is Double.MIN_VALUE, the same value is returned.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.abs Method (Int32)

Returns the absolute value of an int parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int abs(
    int a);
```

Parameters

a

An int expression.

Return Value

The absolute value of the parameter.

Example

The following example displays some int values and their absolutes values.

```
// m-absint.jsl
// Math.abs example using int variables

public class MyClass
{
    public static void main(String[] args)
    {
        int x = Integer.MAX_VALUE;
        int y = Integer.MIN_VALUE;
        int z = -88;

// Original values:
        System.out.println("The original values:");
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);

// Absolute values:
        System.out.println("The absolute values:");
        System.out.println(Math.abs(x));
        System.out.println(Math.abs(y));
        System.out.println(Math.abs(z));
    }
}

/*
The original values:
2147483647
-2147483648
-88
The absolute values:
2147483647
-2147483648
88
*/
```

Remarks

If the parameter is not negative, the parameter is returned.

If the parameter is negative, the negation of the parameter is returned.

If the parameter is Integer.MIN_VALUE, which is equal to -2147483648, the same value is returned.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.abs Method (Int64)

Returns the absolute value of long parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static long abs(
    long a);
```

Parameters

a

A long expression.

Return Value

The absolute value of the parameter.

Example

The following example displays some long values and their absolutes values

```
// m-abslong.jsl
// Math.abs example using long variables

public class MyClass
{
    public static void main(String[] args)
    {
        long x = Long.MAX_VALUE;
        long y = Long.MIN_VALUE;
        long z = -88;

// Original values:
        System.out.println("The original values:");
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);

// Absolute values:
        System.out.println("The absolute values:");
        System.out.println(Math.abs(x));
        System.out.println(Math.abs(y));
        System.out.println(Math.abs(z));
    }
}

/*
The original values:
9223372036854775807
-9223372036854775808
-88
The absolute values:
9223372036854775807
-9223372036854775808
88
*/
```

Remarks

If the parameter is not negative, the parameter is returned.

If the parameter is negative, the negation of the parameter is returned.

If the parameter is Long.MIN_VALUE, which is equal to -9223372036854775808, the same value is returned.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.abs Method (Single)

Returns the absolute value of float parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static float abs(
    float a);
```

Parameters

a

A float expression.

Return Value

The absolute value of the parameter.

Example

The following example displays some float values and their absolutes values

```
// m-absfloat.jsl
// Math.abs example using float variables

public class MyClass
{
    public static void main(String[] args)
    {
        float x = Float.MAX_VALUE;
        float y = Float.MIN_VALUE;
        float z = -88.1f;

        // Original values:
        System.out.println("The original values:");
        System.out.println(x);
        System.out.println(y);
        System.out.println(z);

        // Absolute values:
        System.out.println("The absolute values:");
        System.out.println(Math.abs(x));
        System.out.println(Math.abs(y));
        System.out.println(Math.abs(z));
    }
}

/*
Output:
The original values:
3.40282347E38
1.401298E-45
-88.1
The absolute values:
3.40282347E38
1.401298E-45
88.1
*/
```

Remarks

If the parameter is not negative, the parameter is returned.

If the parameter is negative, the negation of the parameter is returned.

If the parameter is Float.MIN_VALUE, the same value is returned.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.acos Method

Returns the arc cosine of a double value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double acos(
    double a);
```

Parameters

a

A double expression that represents the angle for which the arc cosine is computed.

Return Value

The arc cosine of the double parameter.

Example

The following example starts by converting the angle 60 degrees to radians, and then computes the tangent and arc tangent for this angle.

```
// m-cos1.jsl
// Math.cos and Math.acos example

public class MyClass
{
    public static void main(String[] args)
       {
           double radians = Math.toRadians(60.0);
           System.out.println("The angle in radians =  "+ radians);
           System.out.println("The cosine of the angle = " +
                              Math.cos(radians));
           System.out.println("The angle computed with arc cosine = " +
                              Math.acos(Math.cos(radians)));
       }
}

/*
Output:
The angle in radians =  1.0471975511965976
The cosine of the angle = 0.50000000000000011
The angle computed with arc cosine = 1.0471975511965976
*/
```

Remarks

If the value of the angle is NaN or greater than 1, the returned value will be NaN.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.asin Method

Returns the arc sine of a double value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double asin(
    double a);
```

Parameters

a

A double expression that represents the angle for which the arc sine is calculated.

Return Value

The arc sine of double parameter.

Example

This example displays the sine of the angle PI/2 (90 degrees), which is 1.0, and the arc sine of 1.0 in degrees, which is 90.

```
// m-asin1.jsl
// Math.asin and Math.sin examples
public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("Sine of PI/2 = " + Math.sin(Math.PI/2));
        System.out.println("Arc sine of 1.0 = " +
                    Math.toDegrees(Math.asin(Math.sin(Math.PI/2))));
    }
}

/*
Output:
Sine of PI/2 = 1.0
Arc sine of 1.0 = 90.0
*/
```

Remarks

If the value of the parameter is 0, the returned value will be 0 with the same sign.

If the value of the parameter is NaN or greater than 1, the returned value will be NaN.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.atan Method

Returns the arc tangent of double value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double atan(
    double a);
```

Parameters

a

A double expression that represents the angle for which the arc tangent is calculated.

Return Value

The arc tangent of double parameter.

Example

The following example starts by converting the angle 45 degrees to radians, and then computes the tangent and arc tangent for this angle.

```
// m-atan1.jsl
// Math.tan and Math.atan example

public class MyClass
{
    public static void main(String[] args)
    {
        double radians = Math.toRadians(45.0);
        System.out.println("The angle 45 in radians =  "+ radians);
        System.out.println("The tangent of the angle = " +
                           Math.ceil(Math.tan(radians)));
        System.out.println("The angle computed with atan = " +
                  Math.toDegrees(Math.atan(Math.tan(radians))));
    }
}

/*
Output:
The angle 45 in radians =  0.78539816339744828
The tangent of the angle = 1.0
The angle computed with atan = 45.0
*/
```

Remarks

If the value of the parameter is 0, the returned value will be 0 with the same sign.

If the value of the parameter is NaN, the returned value will be NaN.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.atan2 Method

Converts the Cartesian coordinates of a point (x,y) to the polar coordinates by computing the arc tangent for y/x.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double atan2(
    double y,
    double x);
```

Parameters

y

A double expression that represents the ordinate.

x

A double expression that represents the ordinate the abscissa.

Return Value

The arc tangent for y/x.

Example

This example computes the angle whose tangent is 100.0/100.0. The result is displayed in both radians and degrees.

```
// m-atan2.jsl
// Math.atan2 example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("Angle in radinas computed with atan2 = " +
                           Math.atan2(100.0,100.0));
        System.out.println("Angle in degrees computed with atan2 = " +
                           Math.atan2(100.0, 100.0)*180/Math.PI);
    }
}

/*
Output:
Angle in radinas computed with atan2 = 0.78539816339744828
Angle in degrees computed with atan2 = 45.0
*/
```

Remarks

If the value of one of the parameters is NaN, the returned value will be NaN.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.ceil Method

Returns a double number that represents the smallest mathematical integer that is not less than a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double ceil(
    double a);
```

Parameters

a

A double expression.

Return Value

A double number that represents the smallest mathematical integer that is not less than the double parameter.

Example

```
// m-ceil1.jsl
// Math.ceil example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The ceil of 25.64 = " +
                           Math.ceil(25.64));
    }
}

/*
Output:
The ceil of 25.64 = 26.0
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.cos Method

Returns the cosine of double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double cos(
    double a);
```

Parameters

a

A double expression that represents the angle in radians.

Return Value

The cosine of double parameter.

Example

See the example for acos.

Remarks

If the value of the angle is NaN or infinity, the returned value will be NaN.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.exp Method

Returns the value of the base of the natural system of logarithms (e) raised to the power of a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double exp(
    double a);
```

Parameters

a

A double expression that represents the exponent.

Return Value

The value of "e" raised to the power of the double parameter

Example

This example displays the value of e, which is approximately 2.7183, and the value of e raised to the power zero, which is 1.0.

```
// m-exp1.jsl
// Math.exp example
public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The value of e = " + Math.exp(1.0));
        System.out.println("The value of e raised to the power 0 = " +
                           Math.exp(0));
    }
}

/*
Output:
The value of e = 2.7182818284590451
The value of e raised to the power 0 = 1.0
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.floor Method

Returns a double number that represents the largest mathematical integer that is not greater than a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double floor(
    double a);
```

Parameters

a

A double expression.

Return Value

A double number that represents the largest mathematical integer that is not greater than the double parameter.

Example

```
// m-floor1.jsl
// Math.floor example
public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The floor of 25.64 = " +
                           Math.floor(25.64));
    }
}

/*
Output:
The floor of 25.64 = 25.0
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.IEEEremainder Method

Returns the remainder of the integer division of a double parameter by a second double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double IEEEremainder(
    double f1,
    double f2);
```

Parameters

f1

A double expression that represents the dividend.

f2

A double expression that represents the divisor.

Return Value

The remainder of f1/f2.

Example

The following example calculates the remainder of the division 26.0/4.0.

```
// m-remainder1.jsl
// Math.IEEEremainder example
public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The remainder of 26.0/4.0 = " +
                           Math.IEEEremainder(26.0,4.0));
    }
}

/*
Output:
The remainder of 26.0/4.0 = 2.0
*/
```

Remarks

The remainder is calculated as the result of the operation f1-f2*n, where n the result of the integer division of f1/f2.

If one of the parameters is NaN the result will be NaN.

If the first parameter is infinite or if the second parameter is 0 result will be NaN.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.log Method

Returns the natural logarithm of a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double log(
    double a);
```

Parameters

a

A double expression for which the logarithm is computed.

Return Value

The natural logarithm of the double parameter.

Example

This example calculates the logarithm of E.

```
// m-log1.jsl
// Math.log example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The log of e = " +
                           Math.log(Math.E));
    }
}

/*
Output:
The log of e = 1.0
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.max Method

Overload List

Name	Description
Math.max (double, double)	Returns the larger of two double parameters.
Math.max (int, int)	Returns the larger of two int parameters.
Math.max (long, long)	Returns the larger of two long parameters.
Math.max (float, float)	Returns the larger of two float parameters.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.max Method (Double, Double)

Returns the larger of two double parameters.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double max(
double a,
double b);
```

Parameters

a

A double expression.

b

A double expression.

Return Value

The larger of a and b.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.max Method (Int32, Int32)

Returns the larger of two int parameters.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int max(
    int a,
    int b);
```

Parameters

a

An int expression.

b

An int expression.

Return Value

The larger of a and b.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.max Method (Int64, Int64)

Returns the larger of two long parameters.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static long max(
    long a,
    long b);
```

Parameters

a

A long expression.

b

A long expression.

Return Value

The larger of a and b.

Example
Description

This example displays the larger of two long numbers.

Code

```
// m-maxl.jsl
// Math.max example
public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The maximum of the two long numbers = " +
                           Math.max(2345L,2244L));
    }
}

/*
Output:
The maximum of the two long numbers = 2345
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.max Method (Single, Single)

Returns the larger of two float parameters.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static float max(
    float a,
    float b);
```

Parameters

a

A float expression.

b

A float expression.

Return Value

The larger of a and b.

Example

```
// m-maxf.jsl
// Math.max example
public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The maximum of the two floats = " +
                           Math.max(25.02f,25.01f));
    }
}

/*
Output:
The maximum of the two floats = 25.02
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.min Method

Overload List

Name	Description
Math.min (double, double)	Returns the smaller of two double parameters.
Math.min (int, int)	Returns the smaller of two int parameters.
Math.min (long, long)	Returns the smaller of two long meters.
Math.min (float, float)	Returns the smaller of two float parameters.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.min Method (Double, Double)

Returns the smaller of two double parameters.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double min(
    double a,
    double b);
```

Parameters

a

A double expression.

b

A double expression.

Return Value

The smaller of a and b

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.min Method (Int32, Int32)

Returns the smaller of two int parameters.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int min(
    int a,
    int b);
```

Parameters

a

An int expression.

b

An int expression.

Return Value

The smaller of a and b.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.min Method (Int64, Int64)

Returns the smaller of two long meters.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static long min(
    long a,
    long b);
```

Parameters

a

A long expression.

b

A long expression.

Return Value

The smaller of a and b.

Example

The following example displays the minimum of two long numbers.

```
// min1jsl
// Math.min example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The minimum is = " +
                           Math.min(234L, 62345L));
    }
}

/*
Output:
The minimum is = 234
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.min Method (Single, Single)

Returns the smaller of two float parameters.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static float min(
    float a,
    float b);
```

Parameters

a

A float expression.

b

A float expression.

Return Value

The smaller of a and b.

Example
Description

This example displays the smaller of two float numbers.

Code

```
// m-minf.jsl
// Math.min example
public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The minimum of the two floats = " +
                           Math.min(25.02f,25.01f));
    }
}

/*
Output:
The minimum of the two floats = 25.01
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.pow Method

Returns the value of a base double number raised to the power of a double exponent.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double pow(
    double a,
    double b);
```

Parameters

a

A double expression.

b

A double expression that represents the exponent.

Return Value

The value of a raised to the power b.

Example

This example displays the powers of some double expressions.

```
// m-pow1.jsl
// Math.pow example

public class MyClass
{
   public static void main(String[] args)
   {
      double x = .25;
      double y = 4;
      double z = 0;

      // Original values:
      System.out.println("The original values:");
      System.out.println("x = " + x);
      System.out.println("y = " + y);
      System.out.println("z = " + z);

      // Absolute values:
      System.out.println("The powers:");
      System.out.println("pow(x,y)= "+ Math.pow(x,y));
      System.out.println("pow(y,x)= "+ Math.pow(y,x));
      System.out.println("pow(x,z)= "+ Math.pow(x,z));
   }
}

/*
Output:
The original values:
x = 0.25
y = 4.0
z = 0.0
The powers:
pow(x,y)= 0.00390625
pow(y,x)= 1.4142135623730952
pow(x,z)= 1.0
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.random Method

Returns a random double value between 0.0 and 1.0.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static synchronized double random();
```

Return Value

A value between 0.0 and 1.0.

Example

This example displays the closest mathematical integer to a random value between 1.0 and 10.0.

```
// m-random1.jsl
// Math.random example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The generated random number is = " +
                           Math.rint(Math.random()*10));
    }
}

/*
Output:
The generated random number is = 4.0
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.rint Method

Returns a double number that represents the closest integer to a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double rint(
    double a);
```

Parameters

a

A double expression.

Return Value

A double number that represents the closest integer to the double parameter.

Example

This example displays the closest integer value to 25.02.

```
// m-rint1.jsl
// Math.rint example
public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The closest integer value = " +
                           Math.rint(25.02));
    }
}

/*
Output:
The closest integer value = 25.0
*/
```

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.round Method

Overload List

Name	Description
Math.round (double)	Returns the closest long number to a double parameter.
Math.round (float)	Returns the closest int to a float parameter.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.round Method (Double)

Returns the closest long number to a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static long round(
    double a);
```

Parameters

a

A double expression.

Return Value

The closest long number to the double parameter.

Example

This example displays result of rounding the two double values 45.5 and 45.4.

```
// m-round1.jsl
// Math.round example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The rounded long number of 45.5 is = " +
                           Math.round(45.5));
        System.out.println("The rounded long number of 45.4 is = " +
                           Math.round(45.4));
    }
}

/*
Output:
The rounded long number of 45.5 is = 46
The rounded long number of 45.4 is = 45
*/
```

Remarks

If the number contains a fraction, and if it is equal to or greater than 0.5, it is approximated to 1; truncated otherwise.

If the parameter is NaN, the returned value is 0.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.round Method (Single)

Returns the closest int to a float parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int round(
    float a);
```

Parameters

a

A float expression.

Return Value

The closest int to the float parameter.

Example

This example displays result of rounding the two float values 45.5f and 45.4f.

```
// m-round2.jsl
// Math.round example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The closest int to 45.5f is = " +
                           Math.round(45.5f));
        System.out.println("The closest int to 45.4f is = " +
                           Math.round(45.4f));
    }
}

/*
Output:
The closest int to 45.5f is = 46
The closest int to 45.4f is = 45
*/
```

Remarks

If the number contains a fraction, and if it is equal to or greater than 0.5, it is approximated to 1; truncated otherwise.

If the parameter is NaN, the returned value is 0.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.sin Method

Returns the sine of a double angle.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double sin(
    double a);
```

Parameters

a

The angle in radians for which the sine is computed.

Return Value

The sine of the double parameter.

Example

The following example displays the trigonometric sine and arc sine values for some numbers.

```
// m-asin2.jsl
// Math.asin and Math.sin example
public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("Sine of PI/2 = " + Math.sin(Math.PI/2));
        System.out.println("Arc sine of 1.0 = " +
                           Math.asin(Math.sin(Math.PI/2)));
        System.out.println("Sine of -0.0 = " + Math.sin(-0.0));
        System.out.println("Arc sine of -0.0 = " + Math.asin(-0.0));
        System.out.println("Sine of NaN = " + Math.sin(Math.sqrt(-4)));
        System.out.println("Sine of infinity = " + Math.sin(Math.PI/0));
    }
}

/*
Output:
Sine of PI/2 = 1.0
Arc sine of 1.0 = 1.5707963267948966
Sine of -0.0 = -0.0
Arc sine of -0.0 = -0.0
Sine of NaN = NaN
Sine of infinity = NaN
*/
```

Remarks

If the value of the angle is 0, the returned value will be 0 with the same sign.

If the value of the angle is NaN or infinity, the returned value will be NaN.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.sqrt Method

Returns the square root of a double parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double sqrt(
    double a);
```

Parameters

a

A double expression.

Return Value

The square root of the double parameter.

Example

This example displays the square root of some double numbers.

```
// m-sqrt.jsl
// Math.sqrt example

public class MyClass
{
    public static void main(String[] args)
    {
        double x= -1.0;
        double y = 16.0;
        double z = 0.0;

        // Original values:
        System.out.println("Numbers:");
        System.out.println("x = " + x);
        System.out.println("y = " + y);
        System.out.println("z = " + z);

        // Square roots:
        System.out.println("Square roots:");
        System.out.println("Sqrt(x) = " + Math.sqrt(x));    //Nan
        System.out.println("Sqrt(sqrt(x)) = "
                   + Math.sqrt(Math.sqrt(x)));    //Nan
        System.out.println("Sqrt(y) = " + Math.sqrt(y));
        System.out.println("Sqrt(z) = " + Math.sqrt(z));
    }
}

/*
Output:
Numbers:
x = -1.0
y = 16.0
z = 0.0
Square roots:
Sqrt(x) = NaN
Sqrt(sqrt(x)) = NaN
Sqrt(y) = 4.0
Sqrt(z) = 0.0
*/
```

Remarks

If the argument is NaN or less than zero, the result is NaN.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.tan Method

Returns the tangent of a double angle.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double tan(
    double a);
```

Parameters

a

A double expression that represents the angle in radians.

Return Value

The tangent of the double parameter.

Example

See the example for atan.

Remarks

If the value of the angle is 0, the returned value will be 0 with the same sign.

If the value of the angle is NaN or infinity, the returned value will be NaN.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.toDegrees Method

Converts an angle from radians angle to degrees.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double toDegrees(
    double angrad);
```

Parameters

angrad

A double expression that represents the angle in radians.

Return Value

The angle in degrees.

Example

The following example does two conversion operations, one to convert the angle 45 to radians, and another to convert it back to degrees.

```
// m-toDegrees1.jsl
// Math.toDegrees s example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The angle in degrees = " +
                           Math.toDegrees(Math.toRadians(45.0)));
    }
}

/*
Output:
The angle in degrees = 45.0
*/
```

Remarks

The method returns the result of the expression: angrad * 180/Math.PI.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.toRadians Method

Converts an angle from degrees to radians.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static double toRadians(
    double angdeg);
```

Parameters

angdeg

A double expression that represents the angle in degrees.

Return Value

The angle converted to radians.

Example

This example converts the angle 45 degrees to radians and displays the result.

```
// m-toRadians1.jsl
// Math.toRadins example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println("The angle 45 in radians = " +
                           Math.toRadians(45.0));
    }
}

/*
Output:
The angle 45 in radians = 0.78539816339744828
*/
```

Remarks

The method returns the result of the expression: angdeg * Math.PI/180.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

Math.toString Method

Returns the string representation of an object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```

```

Return Value

The string representation of the object.

Example

```
See the examples on different members.
```

Remarks

This method is implicitly included into the print and prinln statements.

See Also

Reference

Math Class

Concepts

Math Members

java.lang Package

NegativeArraySizeException Class

The exception that is thrown when attempting to create an array with less than zero elements.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.NegativeArraySizeException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.NegativeArraySizeException

See Also

Concepts

NegativeArraySizeException Members

java.lang Package

NegativeArraySizeException Members

The exception that is thrown when attempting to create an array with less than zero elements.

The following tables list the members exposed by the NegativeArraySizeException type.

Public Constructors

Name	Description
NegativeArraySizeException	Overloaded. Initializes a new instance of a NegativeArraySizeException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NegativeArraySizeException Class

Concepts

java.lang Package

NegativeArraySizeException Constructor

Initializes a new instance of a NegativeArraySizeException object.

Overload List

Name	Description
NegativeArraySizeException ()	Initializes a new instance of a NegativeArraySizeException object.
NegativeArraySizeException (String)	Initializes a new instance of a NegativeArraySizeException object with the given message.
NegativeArraySizeException (SerializationInfo, StreamingContext)	Initializes a new instance of a NegativeArraySizeException object during deserialization.
NegativeArraySizeException (String, Exception)	Initializes a new instance of a NegativeArraySizeException object with the given message and inner exception.

See Also

Reference

NegativeArraySizeException Class

Concepts

NegativeArraySizeException Members

java.lang Package

NegativeArraySizeException Constructor ()

Initializes a new instance of a NegativeArraySizeException object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NegativeArraySizeException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NegativeArraySizeException Class

Concepts

NegativeArraySizeException Members

java.lang Package

NegativeArraySizeException Constructor (String)

Initializes a new instance of a NegativeArraySizeException object with the given message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NegativeArraySizeException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
NegativeArraySizeException Class
Concepts
NegativeArraySizeException Members
java.lang Package

NegativeArraySizeException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a NegativeArraySizeException object during deserialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.NegativeArraySizeException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
NegativeArraySizeException Class
Concepts
NegativeArraySizeException Members
java.lang Package

NegativeArraySizeException Constructor (String, Exception)

Initializes a new instance of a NegativeArraySizeException object with the given message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NegativeArraySizeException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the NegativeArraySizeException being thrown.

See Also
Reference
NegativeArraySizeException Class
Concepts
NegativeArraySizeException Members
java.lang Package

NegativeArraySizeException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NegativeArraySizeException Class

Concepts

java.lang Package

NegativeArraySizeException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NegativeArraySizeException Class

Concepts

java.lang Package

NoClassDefFoundError Class

An error occurs when a class fails to load because no definition of this class was found.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.NoClassDefFoundError
    extends java.lang.LinkageError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.NoClassDefFoundError

See Also

Concepts

NoClassDefFoundError Members

java.lang Package

NoClassDefFoundError Members

An error occurs when a class fails to load because no definition of this class was found.

The following tables list the members exposed by the NoClassDefFoundError type.

Public Constructors

Name	Description
NoClassDefFoundError	Overloaded. Constructs a NoClassDefFoundError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a NoClassDefFoundError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoClassDefFoundError Class

Concepts

java.lang Package

NoClassDefFoundError Constructor

Constructs a NoClassDefFoundError object.

Overload List

Name	Description
NoClassDefFoundError ()	Constructs a NoClassDefFoundError object without a text message.
NoClassDefFoundError (String)	Constructs a NoClassDefFoundError object with a specific text message.
NoClassDefFoundError (SerializationInfo, StreamingContext)	Constructs a NoClassDefFoundError object during serialization.
NoClassDefFoundError (String, Exception)	Constructs a NoClassDefFoundError object with a text message and inner exception.

See Also

Reference

NoClassDefFoundError Class

Concepts

NoClassDefFoundError Members

java.lang Package

NoClassDefFoundError Constructor ()

Constructs a NoClassDefFoundError object without a text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoClassDefFoundError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NoClassDefFoundError Class

Concepts

NoClassDefFoundError Members

java.lang Package

NoClassDefFoundError Constructor (String)

Constructs a NoClassDefFoundError object with a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoClassDefFoundError(
    java.lang.String s);
```

Parameters

s

The string that contains the description of the error.

See Also

Reference

NoClassDefFoundError Class

Concepts

NoClassDefFoundError Members

java.lang Package

NoClassDefFoundError Constructor (SerializationInfo, StreamingContext)

Constructs a NoClassDefFoundError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.NoClassDefFoundError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

NoClassDefFoundError Class

Concepts

NoClassDefFoundError Members

java.lang Package

NoClassDefFoundError Constructor (String, Exception)

Constructs a NoClassDefFoundError object with a text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoClassDefFoundError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to NoClassDefFoundError.

See Also

Reference

NoClassDefFoundError Class

Concepts

NoClassDefFoundError Members

java.lang Package

NoClassDefFoundError Methods

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a NoClassDefFoundError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoClassDefFoundError Class

Concepts

java.lang Package

NoClassDefFoundError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NoClassDefFoundError Class

Concepts

java.lang Package

NoSuchFieldError Class

An error thrown when the application tries to access a field of an object and that field does not exist anymore or has changed.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.NoSuchFieldError
    extends java.lang.IncompatibleClassChangeError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.IncompatibleClassChangeError

java.lang.NoSuchFieldError

See Also

Concepts

NoSuchFieldError Members

java.lang Package

NoSuchFieldError Members

An error thrown when the application tries to access a field of an object and that field does not exist anymore or has changed.

The following tables list the members exposed by the NoSuchFieldError type.

Public Constructors

Name	Description
NoSuchFieldError	Overloaded. Constructs a NoSuchFieldError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a NoSuchFieldError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchFieldError Class

Concepts

java.lang Package

NoSuchFieldError Constructor

Constructs a NoSuchFieldError object.

Overload List

Name	Description
NoSuchFieldError ()	Constructs a NoSuchFieldError object without a specific text message.
NoSuchFieldError (String)	Constructs a NoSuchFieldError object with a specific text message.
NoSuchFieldError (SerializationInfo, StreamingContext)	Constructs a NoSuchFieldError object during serialization.
NoSuchFieldError (String, Exception)	Constructs a NoSuchFieldError object with a specific text message and inner exception.

See Also

Reference

NoSuchFieldError Class

Concepts

NoSuchFieldError Members

java.lang Package

NoSuchFieldError Constructor ()

Constructs a NoSuchFieldError object without a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchFieldError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NoSuchFieldError Class

Concepts

NoSuchFieldError Members

java.lang Package

NoSuchFieldError Constructor (String)

Constructs a NoSuchFieldError object with a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchFieldError(
    java.lang.String s);
```

Parameters

s

The string that contains the error message details.

See Also

Reference

NoSuchFieldError Class

Concepts

NoSuchFieldError Members

java.lang Package

NoSuchFieldError Constructor (SerializationInfo, StreamingContext)

Constructs a NoSuchFieldError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.NoSuchFieldError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

NoSuchFieldError Class

Concepts

NoSuchFieldError Members

java.lang Package

NoSuchFieldError Constructor (String, Exception)

Constructs a NoSuchFieldError object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchFieldError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to NoSuchFieldError.

See Also

Reference

NoSuchFieldError Class

Concepts

NoSuchFieldError Members

java.lang Package

NoSuchFieldError Methods

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a NoSuchFieldError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchFieldError Class

Concepts

java.lang Package

NoSuchFieldError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NoSuchFieldError Class

Concepts

java.lang Package

NoSuchFieldException Class

An exception thrown to indicate that a class doesn't contain a specific field.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.NoSuchFieldException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.NoSuchFieldException

See Also

Concepts

NoSuchFieldException Members

java.lang Package

NoSuchFieldException Members

An exception thrown to indicate that a class doesn't contain a specific field.

The following tables list the members exposed by the NoSuchFieldException type.

Public Constructors

Name	Description
NoSuchFieldException	Overloaded. Constructs a NoSuchFieldException exception object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchFieldException Class

Concepts

java.lang Package

NoSuchFieldException Constructor

Constructs a NoSuchFieldException exception object.

Overload List

Name	Description
NoSuchFieldException ()	Constructs a NoSuchFieldException exception object without specific text message.
NoSuchFieldException (String)	Constructs a NoSuchFieldException exception object with a specific text message.
NoSuchFieldException (SerializationInfo, StreamingContext)	Constructs a NoSuchFieldException exception object during serialization.
NoSuchFieldException (String, Exception)	Constructs a NoSuchFieldException exception object with a specific text message and inner exception.

See Also

Reference

NoSuchFieldException Class

Concepts

NoSuchFieldException Members

java.lang Package

NoSuchFieldException Constructor ()

Constructs a NoSuchFieldException exception object without specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchFieldException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NoSuchFieldException Class

Concepts

NoSuchFieldException Members

java.lang Package

NoSuchFieldException Constructor (String)

Constructs a NoSuchFieldException exception object with a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchFieldException(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

NoSuchFieldException Class

Concepts

NoSuchFieldException Members

java.lang Package

NoSuchFieldException Constructor (SerializationInfo, StreamingContext)

Constructs a NoSuchFieldException exception object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.NoSuchFieldException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

NoSuchFieldException Class

Concepts

NoSuchFieldException Members

java.lang Package

NoSuchFieldException Constructor (String, Exception)

Constructs a NoSuchFieldException exception object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchFieldException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to NoSuchFieldException.

See Also

Reference

NoSuchFieldException Class

Concepts

NoSuchFieldException Members

java.lang Package

NoSuchFieldException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchFieldException Class

Concepts

java.lang Package

NoSuchFieldException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NoSuchFieldException Class

Concepts

java.lang Package

NoSuchMethodError Class

An error that occurs when an application references a method that does not exist.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.NoSuchMethodError
    extends java.lang.IncompatibleClassChangeError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.IncompatibleClassChangeError

java.lang.NoSuchMethodError

See Also

Concepts

NoSuchMethodError Members

java.lang Package

NoSuchMethodError Members

An error that occurs when an application references a method that does not exist.

The following tables list the members exposed by the NoSuchMethodError type.

Public Constructors

Name	Description
NoSuchMethodError	Overloaded. Constructs a NoSuchMethodError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a NoSuchMethodError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchMethodError Class

Concepts

java.lang Package

NoSuchMethodError Constructor

Constructs a NoSuchMethodError object.

Overload List

Name	Description
NoSuchMethodError ()	Constructs a NoSuchMethodError object without a specific text message.
NoSuchMethodError (String)	Constructs a NoSuchMethodError object with a specific text message.
NoSuchMethodError (SerializationInfo, StreamingContext)	Constructs a NoSuchMethodError object during serialization.
NoSuchMethodError (String, Exception)	Constructs a NoSuchMethodError object with a specific text message and inner exception.

See Also

Reference

NoSuchMethodError Class

Concepts

NoSuchMethodError Members

java.lang Package

NoSuchMethodError Constructor ()

Constructs a NoSuchMethodError object without a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchMethodError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NoSuchMethodError Class

Concepts

NoSuchMethodError Members

java.lang Package

NoSuchMethodError Constructor (String)

Constructs a NoSuchMethodError object with a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchMethodError(
    java.lang.String s);
```

Parameters

s

The string that contains the NoSuchMethodError message text.

See Also
Reference
NoSuchMethodError Class
Concepts
NoSuchMethodError Members
java.lang Package

NoSuchMethodError Constructor (SerializationInfo, StreamingContext)

Constructs a NoSuchMethodError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.NoSuchMethodError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also
Reference
NoSuchMethodError Class
Concepts
NoSuchMethodError Members
java.lang Package

NoSuchMethodError Constructor (String, Exception)

Constructs a NoSuchMethodError object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchMethodError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to NoSuchMethodError.

See Also

Reference

NoSuchMethodError Class

Concepts

NoSuchMethodError Members

java.lang Package

NoSuchMethodError Methods

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a NoSuchMethodError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchMethodError Class

Concepts

java.lang Package

NoSuchMethodError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NoSuchMethodError Class

Concepts

java.lang Package

NoSuchMethodException Class

An exception thrown when an application tries to access a method that doesn't exist in the class.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.NoSuchMethodException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.NoSuchMethodException

See Also

Concepts

NoSuchMethodException Members

java.lang Package

NoSuchMethodException Members

An exception thrown when an application tries to access a method that doesn't exist in the class.

The following tables list the members exposed by the NoSuchMethodException type.

Public Constructors

Name	Description
NoSuchMethodException	Overloaded. Constructs a NoSuchMethodException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchMethodException Class

Concepts

java.lang Package

NoSuchMethodException Constructor

Constructs a NoSuchMethodException object.

Overload List

Name	Description
NoSuchMethodException ()	Constructs a NoSuchMethodException object without a specified text message..
NoSuchMethodException (String)	Constructs a NoSuchMethodException object with a specified text message.
NoSuchMethodException (SerializationInfo, StreamingContext)	Constructs a NoSuchMethodException object during serialization.
NoSuchMethodException (String, Exception)	Constructs a NoSuchMethodException object with a specified message and an inner exception.

See Also

Reference

NoSuchMethodException Class

Concepts

NoSuchMethodException Members

java.lang Package

NoSuchMethodException Constructor ()

Constructs a NoSuchMethodException object without a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchMethodException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NoSuchMethodException Class

Concepts

NoSuchMethodException Members

java.lang Package

NoSuchMethodException Constructor (String)

Constructs a NoSuchMethodException object with a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchMethodException(
    java.lang.String s);
```

Parameters

s

The string that contains the exception text.

See Also

Reference

NoSuchMethodException Class

Concepts

NoSuchMethodException Members

java.lang Package

NoSuchMethodException Constructor (SerializationInfo, StreamingContext)

Constructs a NoSuchMethodException object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.NoSuchMethodException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

NoSuchMethodException Class

Concepts

NoSuchMethodException Members

java.lang Package

NoSuchMethodException Constructor (String, Exception)

Constructs a NoSuchMethodException object with a specified message and an inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NoSuchMethodException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to the NoSuchMethodException.

See Also

Reference

NoSuchMethodException Class

Concepts

NoSuchMethodException Members

java.lang Package

NoSuchMethodException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchMethodException Class

Concepts

java.lang Package

NoSuchMethodException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NoSuchMethodException Class

Concepts

java.lang Package

NullPointerException Class

An exception thrown when you try to use a null value where an object should be used.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.NullPointerException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.NullPointerException

See Also

Concepts

NullPointerException Members

java.lang Package

NullPointerException Members

An exception thrown when you try to use a null value where an object should be used.

The following tables list the members exposed by the NullPointerException type.

Public Constructors

Name	Description
NullPointerException	Overloaded. Constructs a NullPointerException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an exception object to a NullPointerException object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NullPointerException Class

Concepts

java.lang Package

NullPointerException Constructor

Constructs a NullPointerException object.

Overload List

Name	Description
NullPointerException ()	Constructs a NullPointerException object without a specified text message.
NullPointerException (String)	Constructs a NullPointerException object with a specified text message.
NullPointerException (SerializationInfo, StreamingContext)	Constructs a NullPointerException object during serialization.
NullPointerException (String, Exception)	Constructs a NullPointerException object with a specified text message and inner exception.

See Also

Reference

NullPointerException Class

Concepts

NullPointerException Members

java.lang Package

NullPointerException Constructor ()

Constructs a NullPointerException object without a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NullPointerException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NullPointerException Class

Concepts

NullPointerException Members

java.lang Package

NullPointerException Constructor (String)

Constructs a NullPointerException object with a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NullPointerException(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also
Reference
NullPointerException Class
Concepts
NullPointerException Members
java.lang Package

NullPointerException Constructor (SerializationInfo, StreamingContext)

Constructs a NullPointerException object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.NullPointerException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

NullPointerException Class

Concepts

NullPointerException Members

java.lang Package

NullPointerException Constructor (String, Exception)

Constructs a NullPointerException object with a specified text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NullPointerException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to NullPointerException.

See Also

Reference

NullPointerException Class

Concepts

NullPointerException Members

java.lang Package

NullPointerException Methods

Public Methods

Name	Description
<exceptFilter>	Maps an exception object to a NullPointerException object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NullPointerException Class

Concepts

java.lang Package

NullPointerException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NullPointerException Class

Concepts

java.lang Package

Number Class

An abstract class extended by all classes that represent numeric primitive types such as Byte, Short, Integer, Long, Float, and Double. The class contains abstract methods that convert the object value to any of the other numeric types. All these methods are overridden by the numeric subclasses. It also contains methods to convert numeric types to strings and vise versa.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.lang.Number
    extends java.lang.Object
    implements java.io.Serializable
```

Example

```
See examples on classes: Byte, Short, Integer, Long, Float, and Double.
```

Inheritance Hierarchy

java.lang.Object

java.lang.Number

Derived Classes

See Also

Concepts

Number Members

java.lang Package

Number Members

An abstract class extended by all classes that represent numeric primitive types such as Byte, Short, Integer, Long, Float, and Double. The class contains abstract methods that convert the object value to any of the other numeric types. All these methods are overridden by the numeric subclasses. It also contains methods to convert numeric types to strings and vise versa.

The following tables list the members exposed by the Number type.

Public Constructors

Name	Description
Number	Constructs a new object.

Public Methods

Name	Description
byteValue	Returns the value of a specified object as a byte.
doubleValue	Returns the value of a specified object as a double.
equals	(inherited from Object)
floatValue	Returns the value of a specified object as a float.
hashCode	(inherited from Object)
getClass	(inherited from Object)
intValue	Returns the value of a specified object as an int.
longValue	Returns the value of a specified object as a long.
clone	
shortValue	Returns the value of a specified object as a short.
toString	Overridden. Returns the string representation of the numeric value of the object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Number Class

Concepts

java.lang Package

Number Constructor

Constructs a new object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Number();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Number Class

Concepts

Number Members

java.lang Package

Number Methods

Public Methods

Name	Description
byteValue	Returns the value of a specified object as a byte.
doubleValue	Returns the value of a specified object as a double.
equals	(inherited from Object)
floatValue	Returns the value of a specified object as a float.
hashCode	(inherited from Object)
getClass	(inherited from Object)
intValue	Returns the value of a specified object as an int.
longValue	Returns the value of a specified object as a long.
clone	
shortValue	Returns the value of a specified object as a short.
toString	Overridden. Returns the string representation of the numeric value of the object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Number Class

Concepts

java.lang Package

Number.byteValue Method

Returns the value of a specified object as a byte.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte byteValue();
```

Return Value

The value of the specified object as a byte.

Example

See also examples on classes: Byte, Short, Integer, Long, Float, and Double.

```
// n-byteval1.jsl
// Number.byteValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Number n = new Double(123.4);

        System.out.println(n.byteValue());
    }
}

/*
Output:
123
*/
```

Remarks

The conversion might involve rounding or truncation.

See Also

Reference

Number Class

Concepts

Number Members

java.lang Package

Number.clone Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

Example

See examples on classes: Byte, Short, Integer, Long, Float, and Double.

See Also

Reference

Number Class

Concepts

Number Members

java.lang Package

Number.doubleValue Method

Returns the value of a specified object as a double.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract double doubleValue();
```

Return Value

The value of the specified object as a double.

Example

See examples on classes: Byte, Short, Integer, Long, Float, and Double.

Remarks

The conversion might involve rounding.

See Also

Reference

Number Class

Concepts

Number Members

java.lang Package

Number.floatValue Method

Returns the value of a specified object as a float.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract float floatValue();
```

Return Value

The value of the specified object as a float.

Example

See examples on classes: Byte, Short, Integer, Long, Float, and Double.

Remarks

The conversion might involve rounding.

See Also

Reference

Number Class

Concepts

Number Members

java.lang Package

Number.intValue Method

Returns the value of a specified object as an int.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract int intValue();
```

Return Value

The value of the specified object as an int.

Example

See examples on classes: Byte, Short, Integer, Long, Float, and Double.

Remarks

The conversion might involve rounding or truncation.

See Also

Reference

Number Class

Concepts

Number Members

java.lang Package

Number.longValue Method

Returns the value of a specified object as a long.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract long longValue();
```

Return Value

The value of the specified object as a long.

Example

See examples on classes: Byte, Short, Integer, Long, Float, and Double.

Remarks

The conversion might involve rounding or truncation.

See Also

Reference

Number Class

Concepts

Number Members

java.lang Package

Number.shortValue Method

Returns the value of a specified object as a short.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short shortValue();
```

Return Value

The value of the specified object as a short.

Example

See examples on classes: Byte, Short, Integer, Long, Float, and Double.

Remarks

The conversion might involve rounding or truncation.

See Also

Reference

Number Class

Concepts

Number Members

java.lang Package

Number.toString Method

Returns the string representation of the numeric value of the object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

Return Value

The string representation of the numeric value of the object.

Example

See examples on classes: Byte, Short, Integer, Long, Float, and Double.

See Also

Reference

Number Class

Concepts

Number Members

java.lang Package

NumberFormatException Class

An exception thrown when an invalid number format is encountered.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.NumberFormatException
    extends java.lang.IllegalArgumentException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.IllegalArgumentException

java.lang.NumberFormatException

See Also

Concepts

NumberFormatException Members

java.lang Package

NumberFormatException Members

An exception thrown when an invalid number format is encountered.

The following tables list the members exposed by the NumberFormatException type.

Public Constructors

Name	Description
NumberFormatException	Overloaded. Constructs a NumberFormatException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an exception to a NumberFormatException object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NumberFormatException Class

Concepts

java.lang Package

NumberFormatException Constructor

Constructs a NumberFormatException object.

Overload List

Name	Description
NumberFormatException ()	Constructs a NumberFormatException object without a text message.
NumberFormatException (String)	Constructs an instance of a NumberFormatException object with a text message.
NumberFormatException (SerializationInfo, StreamingContext)	Constructs a NumberFormatException object during serialization.
NumberFormatException (String, Exception)	Constructs a NumberFormatException object with a text message and inner exception.

See Also

Reference

NumberFormatException Class

Concepts

NumberFormatException Members

java.lang Package

NumberFormatException Constructor ()

Constructs a NumberFormatException object without a text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NumberFormatException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NumberFormatException Class

Concepts

NumberFormatException Members

java.lang Package

NumberFormatException Constructor (String)

Constructs an instance of a NumberFormatException object with a text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NumberFormatException(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

NumberFormatException Class

Concepts

NumberFormatException Members

java.lang Package

NumberFormatException Constructor (SerializationInfo, StreamingContext)

Constructs a NumberFormatException object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.NumberFormatException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

NumberFormatException Class

Concepts

NumberFormatException Members

java.lang Package

NumberFormatException Constructor (String, Exception)

Constructs a NumberFormatException object with a text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.NumberFormatException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to NumberFormatException.

See Also

Reference

NumberFormatException Class

Concepts

NumberFormatException Members

java.lang Package

NumberFormatException Methods

Public Methods

Name	Description
<exceptFilter>	Maps an exception to a NumberFormatException object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NumberFormatException Class

Concepts

java.lang Package

NumberFormatException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NumberFormatException Class

Concepts

java.lang Package

OutOfMemoryError Class

An error that occurs when your application runs out of memory.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.OutOfMemoryError
    extends java.lang.VirtualMachineError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.VirtualMachineError

java.lang.OutOfMemoryError

com.ms.com.ComError

See Also

Concepts

OutOfMemoryError Members

java.lang Package

OutOfMemoryError Members

An error that occurs when your application runs out of memory.

The following tables list the members exposed by the OutOfMemoryError type.

Public Constructors

Name	Description
OutOfMemoryError	Overloaded. Constructs a OutOfMemoryError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a OutOfMemoryError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

OutOfMemoryError Class

Concepts

java.lang Package

OutOfMemoryError Constructor

Constructs a OutOfMemoryError object.

Overload List

Name	Description
OutOfMemoryError ()	Constructs a OutOfMemoryError object without a text message.
OutOfMemoryError (String)	Constructs a OutOfMemoryError object with a text message.
OutOfMemoryError (SerializationInfo, StreamingContext)	Constructs a OutOfMemoryError object during serialization.
OutOfMemoryError (String, Exception)	Constructs a OutOfMemoryError object with a text message and inner exception.

See Also

Reference

OutOfMemoryError Class

Concepts

OutOfMemoryError Members

java.lang Package

OutOfMemoryError Constructor ()

Constructs a OutOfMemoryError object without a text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.OutOfMemoryError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

OutOfMemoryError Class

Concepts

OutOfMemoryError Members

java.lang Package

OutOfMemoryError Constructor (String)

Constructs a OutOfMemoryError object with a text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.OutOfMemoryError(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

OutOfMemoryError Class

Concepts

OutOfMemoryError Members

java.lang Package

OutOfMemoryError Constructor (SerializationInfo, StreamingContext)

Constructs a OutOfMemoryError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.OutOfMemoryError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

OutOfMemoryError Class

Concepts

OutOfMemoryError Members

java.lang Package

OutOfMemoryError Constructor (String, Exception)

Constructs a OutOfMemoryError object with a text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.OutOfMemoryError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to OutOfMemoryError.

See Also
Reference
OutOfMemoryError Class
Concepts
OutOfMemoryError Members
java.lang Package

OutOfMemoryError Methods

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a OutOfMemoryError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

OutOfMemoryError Class

Concepts

java.lang Package

OutOfMemoryError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

OutOfMemoryError Class

Concepts

java.lang Package

Process Class

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.lang.Process
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.lang.Process

See Also

Concepts

Process Members

java.lang Package

Process Members

The following tables list the members exposed by the Process type.

Public Constructors

Name	Description
Process	

Public Methods

Name	Description
destroy	
equals	(inherited from Object)
exitValue	
getErrorStream	
hashCode	(inherited from Object)
getInputStream	
getOutputStream	
getClass	(inherited from Object)
clone	
toString	Overridden.
waitFor	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Process Class

Concepts

java.lang Package

Process Constructor

Initializes a new instance of the Process Class .

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Process();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Process Class

Concepts

Process Members

java.lang Package

Process Methods

Public Methods

Name	Description
destroy	
equals	(inherited from Object)
exitValue	
getErrorStream	
hashCode	(inherited from Object)
getInputStream	
getOutputStream	
getClass	(inherited from Object)
clone	
toString	Overridden.
waitFor	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Process Class

Concepts

java.lang Package

Process.clone Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

See Also

Reference

Process Class

Concepts

Process Members

java.lang Package

Process.destroy Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract void destroy();
```

See Also

Reference

Process Class

Concepts

Process Members

java.lang Package

Process.exitValue Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract int exitValue();
```

See Also

Reference

Process Class

Concepts

Process Members

java.lang Package

Process.getErrorStream Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract java.io.InputStream getErrorStream();
```

See Also

Reference

Process Class

Concepts

Process Members

java.lang Package

Process.getInputStream Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract java.io.InputStream getInputStream();
```

See Also

Reference

Process Class

Concepts

Process Members

java.lang Package

Process.getOutputStream Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract java.io.OutputStream getOutputStream();
```

See Also

Reference

Process Class

Concepts

Process Members

java.lang Package

Process.toString Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

See Also

Reference

Process Class

Concepts

Process Members

java.lang Package

Process.waitFor Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract int waitFor() throws java.lang.InterruptedException;
```

See Also

Reference

Process Class

Concepts

Process Members

java.lang Package

Runnable Interface

An interface that abstracts an object that executes code when it is active such as a thread. A class that implements Runnable can run without inheriting the Thread class.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public interface java.lang.Runnable
```

See Also

Concepts

Runnable Members

java.lang Package

Runnable Members

An interface that abstracts an object that executes code when it is active such as a thread. A class that implements Runnable can run without inheriting the Thread class.

The following tables list the members exposed by the Runnable type.

Public Methods

Name	Description
run	An object implementing Runnable must define the run method. When the object is used to create a thread the method is called and executed in a separate thread.

See Also

Reference

Runnable Interface

Concepts

java.lang Package

Runnable Methods

Public Methods

Name	Description
run	An object implementing Runnable must define the run method. When the object is used to create a thread the method is called and executed in a separate thread.

See Also

Reference

Runnable Interface

Concepts

java.lang Package

Runnable.run Method

An object implementing Runnable must define the run method. When the object is used to create a thread the method is called and executed in a separate thread.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public abstract void run();
```

Example

In this example, `MyClass` implements the Runnable interface and defines the run method. In main(), two threads are started. Each one displays a massage and then sleeps for 2 milliseconds.

When you run the program, it keeps printing "Hi..." and "Bye!" until you terminate the program execution.

```
// Runnable.run example

public class MyClass implements Runnable
{
    String myString;
    int delayTime;

    MyClass(String myString, int delayTime)
    {
        this.myString = myString;
        this.delayTime = delayTime;
    }

    public void run()
    {
        try
        {
            for (;;)
            {
                System.out.println(myString);
                Thread.sleep(delayTime);
            }
        }
        catch (InterruptedException e)
        {
            // ...
        }
    }

    public static void main(String[] args)
    {
        Runnable proc1 = new MyClass("Hi... ",2);
        Runnable proc2 = new MyClass("Bye! ",2);

        new Thread(proc1,"thread1").start();
        new Thread(proc2,"thread2").start();
    }
}
```

See Also

Reference

Runnable Interface

Concepts

Runnable Members

java.lang Package

Runtime Class (J#)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.Runtime
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.lang.Runtime

See Also

Concepts

Runtime Members

java.lang Package

Runtime Members

The following tables list the members exposed by the Runtime type.

Public Methods

Name	Description
equals	(inherited from Object)
exec	Overloaded.
exit	
gc	
hashCode	(inherited from Object)
getLocalizedInputStream	
getLocalizedOutputStream	
getRuntime	
getClass	(inherited from Object)
loadLibrary	
clone	
runFinalization	
runfinalizersOnExit	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Runtime Class

Concepts

java.lang Package

Runtime Methods

Public Methods

Name	Description
equals	(inherited from Object)
exec	Overloaded.
exit	
gc	
hashCode	(inherited from Object)
getLocalizedInputStream	
getLocalizedOutputStream	
getRuntime	
getClass	(inherited from Object)
loadLibrary	
clone	
runFinalization	
runfinalizersOnExit	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Runtime Class

Concepts

java.lang Package

Runtime.clone Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.exec Method

Overload List

Name	Description
Runtime.exec (String)	
Runtime.exec (String[])	
Runtime.exec (String, String[])	
Runtime.exec (String[], String[])	

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.exec Method (String)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Process exec(
    java.lang.String command) throws java.io.IOException;
```

Parameters

command

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.exec Method (String[ ])

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Process exec(
    java.lang.String[] cmd) throws java.io.IOException;
```

Parameters

cmd

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.exec Method (String, String[ ])

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Process exec(
    java.lang.String command,
    java.lang.String[] envp) throws java.io.IOException;
```

Parameters

command

envp

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.exec Method (String[ ], String[ ])

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Process exec(
    java.lang.String[] cmd,
    java.lang.String[] envp) throws java.io.IOException;
```

Parameters

cmd

envp

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.exit Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public void exit(
    int status);
```

Parameters

status

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.gc Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public void gc();
```

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.getLocalizedInputStream Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.io.InputStream getLocalizedInputStream(
    java.io.InputStream in);
```

Parameters

in

See Also
Reference
Runtime Class
Concepts
Runtime Members
java.lang Package

Runtime.getLocalizedOutputStream Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.io.OutputStream getLocalizedOutputStream(
    java.io.OutputStream out);
```

Parameters

out

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.getRuntime Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Runtime getRuntime();
```

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.loadLibrary Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized void loadLibrary(
    java.lang.String libname);
```

Parameters

libname

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.runFinalization Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public void runFinalization();
```

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.runfinalizersOnExit Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void runFinalizersOnExit(
    boolean on);
```

Parameters

on

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

Runtime.toString Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

See Also

Reference

Runtime Class

Concepts

Runtime Members

java.lang Package

RuntimeException Class

An exception thrown to indicate that a runtime error has occurred.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.RuntimeException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

Derived Classes

See Also

Concepts

RuntimeException Members

java.lang Package

RuntimeException Members

An exception thrown to indicate that a runtime error has occurred.

The following tables list the members exposed by the RuntimeException type.

Public Constructors

Name	Description
RuntimeException	Overloaded. Constructs a RuntimeException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an exception object to a RuntimeException object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

RuntimeException Class

Concepts

java.lang Package

RuntimeException Constructor

Constructs a RuntimeException object.

Overload List

Name	Description
RuntimeException ()	Constructs a RuntimeException object without a specified text message.
RuntimeException (String)	Constructs a RuntimeException object with a specified text message.
RuntimeException (SerializationInfo, StreamingContext)	Constructs a RuntimeException object during serialization.
RuntimeException (String, Exception)	Constructs a RuntimeException object with a specific text message and inner exception.

See Also

Reference

RuntimeException Class

Concepts

RuntimeException Members

java.lang Package

RuntimeException Constructor ()

Constructs a RuntimeException object without a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.RuntimeException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

RuntimeException Class

Concepts

RuntimeException Members

java.lang Package

RuntimeException Constructor (String)

Constructs a RuntimeException object with a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.RuntimeException(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

RuntimeException Class

Concepts

RuntimeException Members

java.lang Package

RuntimeException Constructor (SerializationInfo, StreamingContext)

Constructs a RuntimeException object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.RuntimeException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

RuntimeException Class

Concepts

RuntimeException Members

java.lang Package

RuntimeException Constructor (String, Exception)

Constructs a RuntimeException object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.RuntimeException(
    java.lang.String msg,
    System.Exception innerEx);
```

Parameters

msg

The string that contains the message text.

innerEx

The inner exception that led to RuntimeException.

See Also

Reference

RuntimeException Class

Concepts

RuntimeException Members

java.lang Package

RuntimeException Methods

Public Methods

Name	Description
<exceptFilter>	Maps an exception object to a RuntimeException object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

RuntimeException Class

Concepts

java.lang Package

RuntimeException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

RuntimeException Class

Concepts

java.lang Package

SecurityException Class

An exception thrown when a security error occurs.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.SecurityException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.SecurityException

See Also

Concepts

SecurityException Members

java.lang Package

SecurityException Members

An exception thrown when a security error occurs.

The following tables list the members exposed by the SecurityException type.

Public Constructors

Name	Description
SecurityException	Overloaded. Constructs a SecurityException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an exception object to a SecurityException object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SecurityException Class

Concepts

java.lang Package

SecurityException Constructor

Constructs a SecurityException object.

Overload List

Name	Description
SecurityException ()	Constructs a SecurityException object without a specified text message.
SecurityException (String)	Constructs a SecurityException object with a specified text message.
SecurityException (SerializationInfo, StreamingContext)	Constructs a SecurityException object during serialization.
SecurityException (String, Exception)	Constructs a SecurityException object with a specific text message and inner exception.

See Also

Reference

SecurityException Class

Concepts

SecurityException Members

java.lang Package

SecurityException Constructor ()

Constructs a SecurityException object without a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.SecurityException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

SecurityException Class

Concepts

SecurityException Members

java.lang Package

SecurityException Constructor (String)

Constructs a SecurityException object with a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.SecurityException(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

SecurityException Class

Concepts

SecurityException Members

java.lang Package

SecurityException Constructor (SerializationInfo, StreamingContext)

Constructs a SecurityException object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.SecurityException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also
Reference
SecurityException Class
Concepts
SecurityException Members
java.lang Package

SecurityException Constructor (String, Exception)

Constructs a SecurityException object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.SecurityException(
    java.lang.String msg,
    System.Exception innerEx);
```

Parameters

msg

The string that contains the message text.

innerEx

The inner exception that led to SecurityException.

See Also

Reference

SecurityException Class

Concepts

SecurityException Members

java.lang Package

SecurityException Methods

Public Methods

Name	Description
<exceptFilter>	Maps an exception object to a SecurityException object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SecurityException Class

Concepts

java.lang Package

SecurityException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

SecurityException Class

Concepts

java.lang Package

Short Class

Wraps the primitive type short into an object. The class also contains methods for conversion and processing Short objects.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Short
    extends java.lang.Number
    implements System.IConvertible, java.lang.Comparable
```

Remarks

This class implements System.IConvertible.

Inheritance Hierarchy

java.lang.Object

  java.lang.Number

   java.lang.Short

See Also

Concepts

Short Members

java.lang Package

Short Members

Wraps the primitive type short into an object. The class also contains methods for conversion and processing Short objects.

The following tables list the members exposed by the Short type.

Public Constructors

Name	Description
Short	Overloaded.

Public Fields

Name	Description
MAX_VALUE	A constant that stores the maximum value a short can have.
MIN_VALUE	A constant that stores the minimum value a short can have.
TYPE	A readonly value that represents the Class instance of the primitive type short.

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value stored in a Short object.
compareTo	Overloaded.
decode	Decodes a string parameter to a Short object.
doubleValue	Overridden. Returns the value of a Short object converted to double.
equals	Overridden. Compares the current Short object to another object.
floatValue	Overridden. Returns value of a Short object converted to float.
hashCode	Overridden. Returns the hash code of a Short object.
getClass	(inherited from Object)
intValue	Overridden. Returns the value of a Short object as an int.
longValue	Overridden. Returns the value of a Short object as a long number.
clone	(inherited from Number)
parseShort	Overloaded.
shortValue	Overridden. Returns the value of a Short object as a short number.
ToBoolean	
ToByte	
ToChar	

ToDateTime	
ToDecimal	
ToDouble	
ToInt16	
ToInt32	
ToInt64	
ToSByte	
ToSingle	
toString	Returns the string that represents a short value.
toString	Overloaded. Overridden.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Overloaded.

See Also

Reference

Short Class

Concepts

java.lang Package

Short Fields

Public Fields

Name	Description
MAX_VALUE	A constant that stores the maximum value a short can have.
MIN_VALUE	A constant that stores the minimum value a short can have.
TYPE	A readonly value that represents the Class instance of the primitive type short.

See Also

Reference

Short Class

Concepts

java.lang Package

Short.MAX_VALUE Field

A constant that stores the maximum value a short can have.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final short MAX_VALUE;
```

Example

```
// sh-MAX_VALUE.jsl
// Short MAX_VALUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("MAX_VALUE = " + Short.MAX_VALUE);
    }
}

/*
Output:
MAX_VALUE = 32767
*/
```

Remarks

The value of MAX_VALUE is 32767.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.MIN_VALUE Field

A constant that stores the minimum value a short can have.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final short MIN_VALUE;
```

Example

```
// sh-MIN_VALUE.jsl
// Short MIN_VALUE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("MIN_VALUE = " + Short.MIN_VALUE);
    }
}

/*
Output:
MIN_VALUE = -32768
*/
```

Remarks

The value of MIN_VALUE is -32768.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.TYPE Field

A readonly value that represents the Class instance of the primitive type short.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Class TYPE;
```

Example

```
// sh-TYPE.jsl
// Short.TYPE example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("TYPE = " + Short.TYPE);
    }
}

/*
Output:
TYPE = short
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short Constructor

Overload List

Name	Description
Short (short)	Constructs a new Short object that represents a specified short value.
Short (String)	Constructs a new Short object that represents the short value specified by a string parameter.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short Constructor (Int16)

Constructs a new Short object that represents a specified short value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Short(
    short value);
```

Parameters

value

A short expression.

Example

```
// sh-ctor1.jsl
// Short.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short((short)123);
        System.out.print("The short value is: " + sh);
    }
}

/*
Output:
The short value is: 123
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short Constructor (String)

Constructs a new Short object that represents the short value specified by a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Short(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

A string expression.

Example

```
// sh-ctor2.jsl
// Short.#ctor example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short("123");
        System.out.print("The short value is: " + sh);
    }
}

/*
Output:
The short value is: 123
*/
```

Remarks

Throws NumberFormatException if the string could not be parsed to a short value.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short Methods

Public Methods

Name	Description
byteValue	Overridden. Returns the byte value stored in a Short object.
compareTo	Overloaded.
decode	Decodes a string parameter to a Short object.
doubleValue	Overridden. Returns the value of a Short object converted to double.
equals	Overridden. Compares the current Short object to another object.
floatValue	Overridden. Returns value of a Short object converted to float.
hashCode	Overridden. Returns the hash code of a Short object.
getClass	(inherited from Object)
intValue	Overridden. Returns the value of a Short object as an int.
longValue	Overridden. Returns the value of a Short object as a long number.
clone	(inherited from Number)
parseShort	Overloaded.
shortValue	Overridden. Returns the value of a Short object as a short number.
ToBoolean	
ToByte	
ToChar	
ToDateTime	
ToDecimal	
ToDouble	
ToInt16	
ToInt32	
ToInt64	
ToSByte	
ToSingle	

toString	Returns the string that represents a short value.
toString	Overloaded. Overridden.
ToType	
ToUInt16	
ToUInt32	
ToUInt64	
valueOf	Overloaded.

See Also

Reference

Short Class

Concepts

java.lang Package

Short.byteValue Method

Returns the byte value stored in a Short object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte byteValue();
```

Return Value

The byte value stored in the Short object.

Example

```
// sh-bVal1.jsl
// Short.byteValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short("123");
        System.out.print("The byte value is: " + sh.byteValue());
    }
}

/*
Output:
The byte value is: 123
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.compareTo Method

Overload List

Name	Description
Short.compareTo (Object)	Compares the current Short object to another object.
Short.compareTo (Short)	Compares the current Short object to another Short object.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.compareTo Method (Object)

Compares the current Short object to another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

If obj is a Short object, it behaves exactly like compareTo, in which case it returns:

0 if the two objects are identical.

-1 if obj is greater then the current object.

1 if obj is less then the current object.

Example

```
See the example on compareTo.
```

Remarks

Throws.ClassCastException if obj is not of the type Short.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.compareTo Method (Short)

Compares the current Short object to another Short object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Short aShort);
```

Parameters

aShort

The Short object to compare to.

Return Value

0 if the two objects are identical. 1 if the value of aShort is less than the value of the current object. -1 if the value of aShort is greater than the value of the current object.

Example

```
// sh-comp1.jsl
// Short.compareTo example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short("123");
        Short sh1 = new Short("124");

        System.out.println(
                   "The result is: " + sh.compareTo(sh));  // 0
        System.out.println(
                   "The result is: " + sh.compareTo(sh1)); // -1
        System.out.println(
                   "The result is: " + sh1.compareTo(sh));  // 1

    }
}

/*
Output:
The result is: 0
The result is: -1
The result is: 1
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.decode Method

Decodes a string parameter to a Short object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Short decode(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be decoded.

Return Value

The Short object that contains the short value of str.

Example

```
// sh-decode1.jsl
// Short.decode example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("0x27");   // Hexadecimal number
        Short sh = Short.decode(s);
        System.out.print("The decoded short is: " + sh);
    }
}

/*
Output:
The decoded short is: 39
*/
```

Remarks

The string parameter str might contain decimal, hexadecimal, or octal digits.

Hexadecimal digits are preceded by "0x", 0X", or "#".

Octal digits are preceded by zero (0).

If the string cannot be parsed to a short value, the exception java.lang.NumberFormatException is thrown.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.doubleValue Method

Returns the value of a Short object converted to double.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double doubleValue();
```

Return Value

The value of the Short object converted to double.

Example

```
// sh-doubleVal1.jsl
// Short.doubleValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short("123");

        // Convert it to double and add it to another double number:
        double d = sh.doubleValue() + 321.4;
        System.out.print("The new double number is: " + d);
    }
}

/*
Output:
The new double number is: 444.4
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.equals Method

Compares the current Short object to another object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

true if the two objects are identical; false otherwise.

Example

```
// sh-equals1.jsl
// Short.Equals example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh1 = new Short("123");
        Short sh2 = new Short("123");
        System.out.print("The result is: " + sh1.Equals(sh2));
    }
}

/*
Output:
The result is: true
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.floatValue Method

Returns value of a Short object converted to float.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float floatValue();
```

Return Value

The value of the Short object converted to float.

Example

```
// sh-floatValue1.jsl
// Short.floatValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short("123");
        // Convert it to float and add it to another float number:
        float f = sh.floatValue() + 321.4F;
        System.out.print("The new float number is: " + f);
    }
}

/*
Output:
The new float number is: 444.4
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.hashCode Method

Returns the hash code of a Short object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code of the Short object.

Example

```
// sh-getHash1.jsl
// Short.GetHashCode example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short("123");
        int h = sh.GetHashCode();
        System.out.print("The hash code is: " + h);
    }
}

/*
Output:
The hash code is: 123
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.intValue Method

Returns the value of a dda946ea-2660-45de-922b-686e3f018370 object as an int.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int intValue();
```

Return Value

The value of the Short object as an int.

Example

```
// sh-intValue1.jsl
// Short.intValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short("123");

        // Convert it and add it to another int:
        int i = sh.intValue() + 321;
        System.out.print("The new int is: " + i);
    }
}

/*
Output:
The new int is: 444
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.longValue Method

Returns the value of a Short object as a long number.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long longValue();
```

Return Value

The value of the Short object as a long number.

Example

```
// sh-longValue1.jsl
// Short.longValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short("123");

        // Convert it and add it to another long number:
        long l = sh.longValue() + 321L;
        System.out.print("The new long number is: " + l);
    }
}

/*
Output:
The new long number is: 444
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.parseShort Method

Overload List

Name	Description
Short.parseShort (String)	Returns the signed short value represented by a string parameter.
Short.parseShort (String, int)	Returns the signed short value represented by a string parameter in the specified radix.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.parseShort Method (String)

Returns the signed short value represented by a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static short parseShort(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

Return Value

The short value represented by str.

Example

```
// sh-parse1.jsl
// Short.parseShort example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("-123");
        short sh = Short.parseShort(s);
        System.out.print("The short number is: " + sh);
    }
}

/*
Output:
The short number is: -123
*/
```

Remarks

The string characters must all be decimal digits except the first character, which can be a minus sign (-).

The returned value is a signed short in the decimal radix.

If the string could not be parsed to a signed decimal short, the exception java.lang.NumberFormatException is thrown.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.parseShort Method (String, Int32)

Returns the signed short value represented by a string parameter in the specified radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static short parseShort(
    java.lang.String str,
    int radix) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

radix

The radix used in parsing str.

Return Value

The signed short value represented by str in the specified radix.

Example

```
// sh-parse2.jsl
// Short.parseShort example

public class MyClass
{
    public static void main(String[] args)
    {
        String s1 = new String("127");   // Decimal
        String s2 = new String("-7F");    // Hexadecimal

        short sh1 = Short.parseShort(s1,10);
        short sh2 = Short.parseShort(s2,16);
        System.out.print("sh1 = " + sh1 + "\nsh2 = " + sh2);
    }
}

/*
Output:
sh1 = 127
sh2 = -127
*/
```

Remarks

The string characters must all be digits in the specified radix except the first character, which can be a minus sign (-).

The exception java.lang.NumberFormatException is generated if any of the following conditions occurred:

The string parameter is null.

The length of the string parameter is zero ("").

If the value of the radix is not in the range Character.MIN_RADIX to Character.MAX_RADIX.

The string does not represent a short value.

If any of the characters in the string, except the first character, which can be a minus sign, is not a valid digit in the specified radix.

See Also

Reference
Short Class
Concepts
Short Members
java.lang Package

Short.shortValue Method

Returns the value of a Short object as a short number.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short shortValue();
```

Return Value

The value of the Short object as a short number.

Example

```
// sh-shvalue1.jsl
// Short.shortValue example

public class MyClass
{
    public static void main(String[] args)
    {
        Short s = new Short("123");
        short sh = s.shortValue();
        System.out.print("The short value is: " + sh);
    }
}

/*
Output:
The short value is: 123
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToBoolean Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public boolean ToBoolean(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Byte ToByte(
    System.IFormatProvider provider);
```

Parameters

provider

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToChar Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public char ToChar(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToDateTime Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.DateTime ToDateTime(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToDecimal Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.Decimal ToDecimal(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToDouble Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public double ToDouble(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public short ToInt16(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int ToInt32(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public long ToInt64(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToSByte Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public byte ToSByte(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToSingle Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public float ToSingle(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.toString Method (J#)

Returns the string that represents a short value.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String toString(
    short sval);
```

Parameters

sval

A short expression

Return Value

The string that represents the value of bval.

Example

```
// sh-toStr1.jsl
// Short.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        short sh = 127;
        String s = Short.toString(sh);
        System.out.print("The string is: " + s);
    }
}

/*
The result is: 127
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.toString Method

Overload List

Name	Description
Short.ToString ()	Returns the string that represents the value of a Short object.
Short.ToString (IFormatProvider)	

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.toString Method ()

Returns the string that represents the value of a Short object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The string that represents the value of the Short object.

Example

```
// sh-toStr2.jsl
// Short.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        Short sh = new Short("123");
        // Converting to a string and catenating to another string:
        String s = "The string is: " + sh.toString();
        System.out.print(s);
    }
}

/*
The string is: 123
*/
```

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.toString Method (IFormatProvider)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String ToString(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToType Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object ToType(
    System.Type conversionType,
    System.IFormatProvider provider);
```

Parameters

conversionType

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToUInt16 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt16 ToUInt16(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToUInt32 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt32 ToUInt32(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.ToUInt64 Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public System.UInt64 ToUInt64(
    System.IFormatProvider provider);
```

Parameters

provider

Example

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.valueOf Method

Overload List

Name	Description
Short.valueOf (String)	Returns the Short object represented by a string parameter.
Short.valueOf (String, int)	Returns the Short object represented by a string parameter in the specified radix.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.valueOf Method (String)

Returns the Short object represented by a string parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Short valueOf(
    java.lang.String str) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

Return Value

The Short object represented by str.

Example

```
// sh-ValOf1.jsl
// Short.valueOf example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("-128");
        Short sh = Short.valueOf(s);
        System.out.print("The short value is: " + sh);
    }
}

/*
Output:
The short value is: -128
*/
```

Remarks

The string characters must all be decimal digits except the first character, which can be a minus sign (-).

If the string could not be parsed to a signed decimal short, the exception java.lang.NumberFormatException is thrown.

See Also

Reference

Short Class

Concepts

Short Members

java.lang Package

Short.valueOf Method (String, Int32)

Returns the Short object represented by a string parameter in the specified radix.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Short valueOf(
    java.lang.String str,
    int radix) throws java.lang.NumberFormatException;
```

Parameters

str

The string to be parsed.

radix

The radix used in parsing str.

Return Value

The Short object represented by str in the specified radix.

Example

```
// sh-ValOf2.jsl
// Short.valueOf example

public class MyClass
{
    public static void main(String[] args)
    {
        String s = new String("-80");
        Short sh = Short.valueOf(s,16);
        System.out.print("The short value is: " + sh);
    }
}

/*
Output:
The short value is: -128
*/
```

Remarks

The string characters must all be digits in the specified radix except the first character, which can be a minus sign (-).

The exception java.lang.NumberFormatException is generated if any of the following conditions occurred:

The string parameter is null.

The length of the string parameter is zero ("").

If the value of the radix is not in the range Character.MIN_RADIX to Character.MAX_RADIX.

The string does not represent a short value.

If any of the characters in the string, except the first character, which can be a minus sign, is not a valid digit in the specified radix.

See Also

Reference

Short Class

Concepts

Short Members

StackOverflowError Class

An error that is issued when a stack overflow occurs.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.StackOverflowError
    extends java.lang.VirtualMachineError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.VirtualMachineError

java.lang.StackOverflowError

See Also

Concepts

StackOverflowError Members

java.lang Package

StackOverflowError Members

An error that is issued when a stack overflow occurs.

The following tables list the members exposed by the StackOverflowError type.

Public Constructors

Name	Description
StackOverflowError	Overloaded. Constructs a StackOverflowError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a StackOverflowError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StackOverflowError Class

Concepts

java.lang Package

StackOverflowError Constructor

Constructs a StackOverflowError object.

Overload List

Name	Description
StackOverflowError ()	Constructs a StackOverflowError object without a specific text message.
StackOverflowError (String)	Constructs a StackOverflowError object with a specific text message.
StackOverflowError (SerializationInfo, StreamingContext)	Constructs a StackOverflowError object during serialization.
StackOverflowError (String, Exception)	Constructs a StackOverflowError object with a specific text message and inner exception.

See Also

Reference

StackOverflowError Class

Concepts

StackOverflowError Members

java.lang Package

StackOverflowError Constructor ()

Constructs a StackOverflowError object without a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StackOverflowError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

StackOverflowError Class

Concepts

StackOverflowError Members

java.lang Package

StackOverflowError Constructor (String)

Constructs a StackOverflowError object with a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StackOverflowError(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

StackOverflowError Class

Concepts

StackOverflowError Members

java.lang Package

StackOverflowError Constructor (SerializationInfo, StreamingContext)

Constructs a StackOverflowError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.StackOverflowError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also
Reference
StackOverflowError Class
Concepts
StackOverflowError Members
java.lang Package

StackOverflowError Constructor (String, Exception)

Constructs a StackOverflowError object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StackOverflowError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to StackOverflowError.

See Also

Reference

StackOverflowError Class

Concepts

StackOverflowError Members

java.lang Package

StackOverflowError Methods

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a StackOverflowError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StackOverflowError Class

Concepts

java.lang Package

StackOverflowError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

StackOverflowError Class

Concepts

java.lang Package

StringBuffer Class

Implements a sequence of characters, which can be modified by using the appropriate method calls. The main difference between a StringBuffer object and String object is that the latter is immutable. Therefore, it is recommended that you use StringBuffer to manipulate and update the string, and then convert it to a String object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.StringBuffer
    extends java.lang.Object
    implements java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

java.lang.StringBuffer

See Also

Concepts

StringBuffer Members

java.lang Package

StringBuffer Members

Implements a sequence of characters, which can be modified by using the appropriate method calls. The main difference between a StringBuffer object and String object is that the latter is immutable. Therefore, it is recommended that you use StringBuffer to manipulate and update the string, and then convert it to a String object.

The following tables list the members exposed by the StringBuffer type.

Public Constructors

Name	Description
StringBuffer	Overloaded.

Public Methods

Name	Description
append	Overloaded.
capacity	Returns the current capacity of StringBuffer.
charAt	Returns the character at a specified index.
delete	Deletes the characters within a specified range in a StringBuffer object and returns the new StringBuffer object.
deleteCharAt	Deletes the character at a specified index.
ensureCapacity	Ensures that the buffer capacity is not less than a specified minimum.
equals	(inherited from Object)
getChars	Copies the characters within a specified range from a StringBuffer object to a destination character array at a specified position.
hashCode	(inherited from Object)
getClass	(inherited from Object)
insert	Overloaded.
length	Returns the length of the StringBuffer object.
clone	
replace	Replaces characters within a specified range in a StringBuffer object with a specified string.
reverse	Reverses the sequence of the characters in a StringBuffer object.
setCharAt	Changes a character at a specified index in a StringBuffer object.
setLength	Changes the length of a StringBuffer object to a specified length.
substring	Overloaded.
toString	Overridden. Converts a StringBuffer object to a String object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringBuffer Class

Concepts

java.lang Package

StringBuffer Constructor

Overload List

Name	Description
StringBuffer ()	Constructs a new StringBuffer object.
StringBuffer (int)	Constructs a new StringBuffer object with a specified initial capacity.
StringBuffer (String)	Constructs a new StringBuffer object using a specified string for the initial contents.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer Constructor ()

Constructs a new StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer();
```

Example

```
StringBuffer buf = new StringBuffer();   // Initial capacity 16.
```

Remarks

The default initial capacity of the buffer is 16 characters.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer Constructor (Int32)

Constructs a new StringBuffer object with a specified initial capacity.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer(
    int capacity);
```

Parameters

capacity

The initial capacity of the object.

Example

```
StringBuffer buf = new StringBuffer(30);   // Initial capacity 30.
```

Remarks

Throws NegativeArraySizeException if capacity is negative.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer Constructor (String)

Constructs a new StringBuffer object using a specified string for the initial contents.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer(
    java.lang.String str);
```

Parameters

str

A string that represents the initial contents of the StringBuffer object.

Example

```
// Construct a StringBuffer object using a string:
StringBuffer buf = new StringBuffer("Hi there.");
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer Methods

Public Methods

Name	Description
append	Overloaded.
capacity	Returns the current capacity of StringBuffer.
charAt	Returns the character at a specified index.
delete	Deletes the characters within a specified range in a StringBuffer object and returns the new StringBuffer object.
deleteCharAt	Deletes the character at a specified index.
ensureCapacity	Ensures that the buffer capacity is not less than a specified minimum.
equals	(inherited from Object)
getChars	Copies the characters within a specified range from a StringBuffer object to a destination character array at a s pecified position.
hashCode	(inherited from Object)
getClass	(inherited from Object)
insert	Overloaded.
length	Returns the length of the StringBuffer object.
clone	
replace	Replaces characters within a specified range in a StringBuffer object with a specified string.
reverse	Reverses the sequence of the characters in a StringBuffer object.
setCharAt	Changes a character at a specified index in a StringBuffer object.
setLength	Changes the length of a StringBuffer object to a specified length.
substring	Overloaded.
toString	Overridden. Converts a StringBuffer object to a String object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringBuffer Class

Concepts

java.lang Package

StringBuffer.append Method

Overload List

Name	Description
StringBuffer.append (boolean)	Appends a boolean value to a StringBuffer object.
StringBuffer.append (char)	Appends a character to a StringBuffer object.
StringBuffer.append (char[])	Appends a character array to a StringBuffer object.
StringBuffer.append (double)	Appends the string representation of a double to a StringBuffer object.
StringBuffer.append (int)	Appends the string representation of an int to a StringBuffer object.
StringBuffer.append (long)	Appends the string representation of a long to a StringBuffer object.
StringBuffer.append (Object)	Appends the string representation of an object to a StringBuffer object.
StringBuffer.append (float)	Appends the string representation of a float to the value of a StringBuffer object.
StringBuffer.append (String)	Appends a string to a StringBuffer object.
StringBuffer.append (char[], int, int)	Appends characters from a character array, starting at a specified index, to a StringBuffer object.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (Boolean)

Appends a boolean value to a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer append(
    boolean b);
```

Parameters

b

A Boolean expression.

Return Value

The value of the StringBuffer object after appending the boolean value.

Example

```
// sb-appBool.jsl
// StringBuffer.append example

public class MyClass
{
    public static void main(String[] args)
    {
        // Initialize a boolean variable:
        boolean b = true;

        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("Is that ");

        // Append the boolean value and display the buffer:
        System.out.println(s.append(b) + "?");
    }
}

/*
Output:
Is that true?
*/
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (Char)

Appends a character to a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer append(
    char c);
```

Parameters

c

The character to append.

Return Value

The value of the StringBuffer object after appending the character.

Example

```
// sb-appchar.jsl
// StringBuffer.append example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("Hello there");

        // Append a char and display the buffer:
        System.out.println(s.append('!'));
    }
}

/*
Output:
Hello there!
*/
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (Char[ ])

Appends a character array to a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer append(
    char[] str);
```

Parameters

str

The character array to append.

Return Value

The value of the StringBuffer object after appending the character array.

Example

```
// sb-appca.jsl
// StringBuffer.append example

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare and initialize a char array:
        char[] ca = {'A', 'B', 'C'};

        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer(
                         "To learn a language, start with ");

        // Append the char array and display the buffer:
        System.out.println(s.append(ca) + ".");
    }
}

/*
Output:
To learn a language, start with ABC.
*/
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (Double)

Appends the string representation of a double to a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer append(
    double d);
```

Parameters

d

A double expression.

Return Value

The value of the StringBuffer object after appending the string representation of the double value.

Example

```
// sb-appdbl.jsl
// StringBuffer.append example

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare and initialize a double:
        double d = 3.14;

        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("The ratio is: ");

        // Append the double value and display the buffer:
        System.out.println(s.append(d) + ".");
    }
}

/*
Output:
The ratio is: 3.14.
*/
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (Int32)

Appends the string representation of an int to a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer append(
    int i);
```

Parameters

i

An int expression.

Return Value

The value of the StringBuffer object after appending the string representation of the int value.

Example

See the example on append.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (Int64)

Appends the string representation of a long to a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer append(
    long l);
```

Parameters

l

A long expression.

Return Value

The value of the StringBuffer object after appending the string representation of the long value.

Example

See the example on append.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (Object)

Appends the string representation of an object to a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer append(
    java.lang.Object obj);
```

Parameters

obj

An object.

Return Value

The value of the StringBuffer object after appending the string representation of the object.

Example

```
// sb-appobj.jsl
// StringBuffer.append example

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare and initialize an object:
        Object d = new Double(3.14);

        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("The ratio is: ");

        // Append the object and display the buffer:
        System.out.println(s.append(d) + ".");
    }
}

/*
Output:
The ratio is: 3.14.
*/
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (Single)

Appends the string representation of a float to the value of a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer append(
    float f);
```

Parameters

f

A float expression.

Return Value

The value of the StringBuffer object after appending the string representation of the float value.

Example

See the example on append.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (String)

Appends a string to a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer append(
    java.lang.String str);
```

Parameters

str

Return Value

The value of the StringBuffer object after appending the string.

Example

```
// sb-appstr.jsl
// StringBuffer.append example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a String object:
        String s1 = new String("3.14");

        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("The ratio is: ");

        // Append the string and display the buffer:
        System.out.println(s.append(s1) + ".");
    }
}

/*
Output:
The ratio is: 3.14.
*/
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.append Method (Char[ ], Int32, Int32)

Appends characters from a character array, starting at a specified index, to a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer append(
    char[] str,
    int offset,
    int len);
```

Parameters

str

The name of the char array.

offset

The index of the first character to append.

len

The number characters to append.

Return Value

The value of the StringBuffer object after appending the characters.

Example

```
// sb-appca1.jsl
// StringBuffer.append example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a char array:
        char[] ca = {'A', 'B', 'C'};

        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer(
                         "To learn a language, start with ");

        // Append the char array and display the buffer:
        System.out.println(s.append(ca,0,3) + ".");
    }
}

/*
Output:
To learn a language, start with ABC.
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if len exceeds the number of available characters starting from offset.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.capacity Method

Returns the current capacity of StringBuffer.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int capacity();
```

Return Value

The capacity of StringBuffer.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.charAt Method

Returns the character at a specified index.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized char charAt(
    int index);
```

Parameters

index

The index of the character.

Return Value

The character at index.

Example

```
// sb-charat.jsl
// StringBuffer.charAt example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer(
                       /* 01234567890123456789012345678901234 */
                         "To learn a language, start with J#.");
        int idx = 33;

        // Display the character at index 33:
        System.out.println("The character at the position " +
                           idx + " is: \"" + s.charAt(idx) + "\".");
    }
}

/*
Output:
The character at the position 33 is: "#".
*/
```

Remarks

Throws StringIndexOutOfBoundsException, if index is negative, or greater than or equal to length

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.delete Method

Deletes the characters within a specified range in a StringBuffer object and returns the new StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer delete(
    int start,
    int end);
```

Parameters

start

The starting index.

end

The ending index.

Return Value

The new StringBuffer object.

Example

```
// sb-del.jsl
// StringBuffer.delete example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer(
                       /* 01234567890123456789012345678901 2 */
                         "To learn J#, start with keywords.");
        int start = 0;
        int end = 13;

        // Delete chars from start to end:
        s = s.delete(start,end);

        // Catenate a string and display the buffer:
        System.out.println("Always " + s);
    }
}

/*
Output:
Always start with keywords.
*/
```

Remarks

The characters are deleted from start, up to, but not including end.

The exception StringIndexOutOfBoundsException is thrown, if start is negative, greater than length, or greater than end.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.deleteCharAt Method

Deletes the character at a specified index.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer deleteCharAt(
    int index);
```

Parameters

index

The index of the character to be deleted.

Return Value

The new StringBuffer object.

Example

```
// sb-delCharAt.jsl
// StringBuffer.deleteCharAt example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer(
                       /* 01234567890123456789012345678901 2 */
                         "To learn J#, start with keywords.");
        int index = 32;

        // Delete the char at position 32:
        s = s.deleteCharAt(index);

        // Catenate a string and display the buffer:
        System.out.println(s + "!");
    }
}

/*
Output:
To learn J#, start with keywords!
*/
```

Remarks

Throws StringIndexOutOfBoundsException if index is negative, or greater than or equal the length of StringBuffer.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.ensureCapacity Method

Ensures that the buffer capacity is not less than a specified minimum.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized void ensureCapacity(
    int minimumCapacity);
```

Parameters

minimumCapacity

The minimum capacity.

Example

```
// Ensure that capacity is not less than 32:
s.ensureCapacity(32);
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.getChars Method

Copies the characters within a specified range from a StringBuffer object to a destination character array at a specified position.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized void getChars(
    int srcBegin,
    int srcEnd,
    char[] dst,
    int dstBegin);
```

Parameters

srcBegin

The beginning position of the range.

srcEnd

The ending position of the range.

dst

The destination character array.

dstBegin

The insertion position.

Example

In this example, two characters are copied from the StringBuffer object, src, to a destination character array, dst. The destination array contains the string "J#."

```
// sb-getCh.jsl
// StringBuffer.getChars example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer src = new StringBuffer(
                       /* 012345678901234567890123456789012 */
                         "To learn J#, start with keywords.");

        // Declare a new char array:
        char[] dst = new char[2];

        // Copy the chars #9 and #10 to dst:
        src.getChars(9,11,dst,0);

        // Display dst:
        System.out.println(dst);
    }
}

/*
Output:
J#
*/
```

Remarks

Throws NullPointerException if dst is null.

Throws StringIndexOutOfBoundsException if any of the following conditions is true:

srcBegin is negative.

dstBegin is negative.

srcBegin is greater than srcEnd.

srcEnd is greater than the current length of the string buffer.

dstBegin + (srcEnd -srcBegin) is greater than dst.length.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.insert Method

Overload List

Name	Description
StringBuffer.insert (int, boolean)	Inserts the string representation of a boolean value into a StringBuffer object at a specified offset.
StringBuffer.insert (int, char)	Inserts the string representation of a char into a StringBuffer object at a specified offset.
StringBuffer.insert (int, char[])	Inserts the string representation of a character array into a StringBuffer object at a specified offset.
StringBuffer.insert (int, double)	Inserts the string representation of a double value into a StringBuffer object at a specified offset.
StringBuffer.insert (int, int)	Inserts the string representation of an int into a StringBuffer object at a specified offset.
StringBuffer.insert (int, long)	Inserts the string representation of a long value into a StringBuffer object at a specified offset.
StringBuffer.insert (int, Object)	Inserts the string representation of an object into a StringBuffer object at a specified offset.
StringBuffer.insert (int, float)	Inserts the string representation of a float into a StringBuffer object at a specified offset.
StringBuffer.insert (int, String)	Inserts the string into a StringBuffer object at a specified offset.
StringBuffer.insert (int, char[], int, int)	Inserts characters from a character array starting at a specified character offset to a specified offset in a StringBuffer object.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.insert Method (Int32, Boolean)

Inserts the string representation of a boolean value into a StringBuffer object at a specified offset.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer insert(
    int offset,
    boolean b);
```

Parameters

offset

The offset of insertion.

b

The boolean value.

Return Value

The same instance of StringBuffer with data inserted.

Example

In this example the string representation of the boolean value true is inserted at the offset 6 of the StringBuffer object.

```
// sb-insert1.jsl
// StringBuffer.insert example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer buf = new StringBuffer(
                        /* 0123456 */
                          "Is it ?");

        // Insert true at offset 6:
        buf = buf.insert(6,true);

        // Display the buffer:
        System.out.println(buf);
    }
}

/*
Output:
Is it true?
*/
```

Remarks

A valid offset must be greater than or equal to zero and less than or equal to the length of StringBuffer.

Throws StringIndexOutOfBoundsException if offset is invalid.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

StringBuffer.insert Method (Int32, Char)

Inserts the string representation of a char into a StringBuffer object at a specified offset.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer insert(
    int offset,
    char c);
```

Parameters

offset

The offset of the insertion.

c

The character to be inserted.

Return Value

The same instance of StringBuffer with data inserted.

Example

In this example the character 'J' is inserted at offset 6 of the StringBuffer object.

```
// sb-insert2.jsl
// StringBuffer.insert example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer buf = new StringBuffer(
                        /* 0123456 */
                          "Hello #!");

        // Insert 'J' at the offset 6:
        buf = buf.insert(6,'J');

        // Display the buffer:
        System.out.println(buf);
    }
}

/*
Output:
Hello J#!
*/
```

Remarks

A valid offset must be greater than or equal to zero and less than or equal to the length of StringBuffer.

Throws StringIndexOutOfBoundsException if offset is invalid.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

StringBuffer.insert Method (Int32, Char[])

Inserts the string representation of a character array into a StringBuffer object at a specified offset.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer insert(
    int offset,
    char[] str);
```

Parameters

offset

The offset of the insertion.

str

The character array to be inserted.

Return Value

The same instance of StringBuffer with data inserted.

Example

```
// sb-insert3.jsl
// StringBuffer.insert example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer buf = new StringBuffer(
                       /* 0123456 */
                         "Hello !");

        // Construct a character array:
        char[] str = {'J','#'};

        // Insert str at the offset 6:
        buf = buf.insert(6,str);

        // Display the buffer:
        System.out.println(buf);
    }
}

/*
Output:
Hello J#!
*/
```

Remarks

A valid offset must be greater than or equal to zero and less than or equal to the length of StringBuffer.

Throws StringIndexOutOfBoundsException if offset is invalid.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

StringBuffer.insert Method (Int32, Double)

Inserts the string representation of a double value into a StringBuffer object at a specified offset.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer insert(
    int offset,
    double d);
```

Parameters

offset

The offset of the insertion.

d

The double value to be inserted.

Return Value

The same instance of StringBuffer with data inserted.

Example

```
// sb-insert4.jsl
// StringBuffer.insert example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer buf = new StringBuffer(
                       /* 0123456 */
                         "Hello !");

        // Construct a Double object:
        Double d = new Double(3.45);

        // Insert d at offset 6:
        buf = buf.insert(6,d);

        // Display the buffer:
        System.out.println(buf);
    }
}

/*
Output:
Hello 3.45!
*/
```

Remarks

A valid offset must be greater than or equal to zero and less than or equal to the length of StringBuffer.

Throws StringIndexOutOfBoundsException if offset is invalid.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

StringBuffer.insert Method (Int32, Int32)

Inserts the string representation of an int into a StringBuffer object at a specified offset.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer insert(
    int offset,
    int i);
```

Parameters

offset

The offset of the insertion.

i

The int to be inserted.

Return Value

The same instance of StringBuffer with data inserted.

Example

See the example on insert.

Remarks

A valid offset must be greater than or equal to zero and less than or equal to the length of StringBuffer.

Throws StringIndexOutOfBoundsException if offset is invalid.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.insert Method (Int32, Int64)

Inserts the string representation of a long value into a StringBuffer object at a specified offset.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer insert(
    int offset,
    long l);
```

Parameters

offset

The offset of the insertion.

l

The long value to be inserted.

Return Value

The same instance of StringBuffer with data inserted.

Example

See the example on insert.

Remarks

A valid offset must be greater than or equal to zero and less than or equal to the length of StringBuffer.

Throws StringIndexOutOfBoundsException if offset is invalid.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.insert Method (Int32, Object)

Inserts the string representation of an object into a StringBuffer object at a specified offset.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer insert(
    int offset,
    java.lang.Object obj);
```

Parameters

offset

The offset of the insertion.

obj

The object to be inserted.

Return Value

The same instance of StringBuffer with data inserted.

Example

```
// sb-insert5.jsl
// StringBuffer.insert example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer buf = new StringBuffer(
                       /* 0123456 */
                         "Hello !");

        // Construct an object:
        Object d = new Double(3.45);

        // Insert d at the offset 6:
        buf = buf.insert(6,d);

        // Display the buffer:
        System.out.println(buf);
    }
}

/*
Output:
Hello 3.45!
*/
```

Remarks

A valid offset must be greater than or equal to zero and less than or equal to the length of StringBuffer.

Throws StringIndexOutOfBoundsException if offset is invalid.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

StringBuffer.insert Method (Int32, Single)

Inserts the string representation of a float into a StringBuffer object at a specified offset.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer insert(
    int offset,
    float f);
```

Parameters

offset

The offset of the insertion.

f

The float to be inserted.

Return Value

The same instance of StringBuffer with data inserted.

Example

See the example on insert.

Remarks

A valid offset must be greater than or equal to zero and less than or equal to the length of StringBuffer.

Throws StringIndexOutOfBoundsException if offset is invalid.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.insert Method (Int32, String)

Inserts the string into a StringBuffer object at a specified offset.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer insert(
    int offset,
    java.lang.String str);
```

Parameters

offset

The offset of the insertion.

str

The string to be inserted.

Return Value

The same instance of StringBuffer with data inserted.

Example

```
// sb-insert6.jsl
// StringBuffer.insert example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer buf = new StringBuffer(
                       /* 0123456 */
                         "Hello !");

        // Construct a String object:
        String s = new String("there");

        // Insert the string "s" at offset 6:
        buf = buf.insert(6,s);

        // Display the buffer:
        System.out.println(buf);
    }
}

/*
Output:
Hello there!
*/
```

Remarks

A valid offset must be greater than or equal to zero and less than or equal to the length of StringBuffer.

Throws StringIndexOutOfBoundsException if offset is invalid.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

StringBuffer.insert Method (Inta32, Char[], Int32, Int32)

Inserts characters from a character array starting at a specified character offset to a specified offset in a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer insert(
    int index,
    char[] data,
    int offset,
    int count);
```

Parameters

index

The index where the insertion starts.

data

The character array.

offset

The offset of the first character in the character array to be inserted.

count

The number of characters to be inserted.

Return Value

The same instance of StringBuffer with data inserted.

Example

```
// sb-insert7.jsl
// StringBuffer.insert example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer buf = new StringBuffer(
                       /* 01234567890123456789012345678901234567890 */
                         "To learn programming, start with keywords.");

        // Construct a char array:
        char[] data = {' ','J','#'};

        // Insert data into index 8 of the StringBuffer object,
        // starting at character 0 with length 4:
        buf.insert(8,data,0,3);

        // Display the buffer:
        System.out.println(buf);
    }
}

/*
Output:
To learn J# programming, start with keywords.
*/
```

Remarks

Throws StringIndexOutOfBoundsException if one of the following conditions is true:

index is negative or greater than the length of StringBuffer.

offset is negative or greater than the data.length.

count is negative or (offset + count) is greater than data.length.

Throws NullPointerException if data is null.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.length Method

Returns the length of the StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public int length();
```

Return Value

The number of characters contained in a StringBuffer object.

Example

```
// sb-length.jsl
// StringBuffer.length example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct two StringBuffer objects:
        StringBuffer s1 = new StringBuffer("Hello world!");
        StringBuffer s2 = new StringBuffer("Hi there.");

        // Display the length of each buffer:
        System.out.println(s1 + " = " + s1.length() + " characters");
        System.out.println(s2 + " = " + s2.length() + " characters");
    }
}

/*
Output:
Hello world! = 12 characters
Hi there. = 9 characters
*/
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.replace Method

Replaces characters within a specified range in a StringBuffer object with a specified string.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer replace(
    int start,
    int end,
    java.lang.String str);
```

Parameters

start

The starting index of the range (inclusive).

end

The ending index of the range (exclusive).

str

The string to replace the characters in the specified range.

Example

```
// sb-replace.jsl
// StringBuffer.insert example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer buf = new StringBuffer(
                       /* 01234567890123456789012345678901234567890 */
                         "To learn programming, start with keywords.");

        // Construct a string:
        String str = new String("about J#");

        // Replace "Programming" with str:
        buf.replace(9,20,str);

        // Display the buffer:
        System.out.println(buf);
    }
}

/*
Output:
To learn about J#, start with keywords.
*/
```

Remarks

Throws StringIndexOutOfBoundsException if start is negative, greater than the length of StringBuffer, or greater than end.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.reverse Method

Reverses the sequence of the characters in a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.StringBuffer reverse();
```

Return Value

Returns the same instance of StringBuffer with data reversed in place.

Example

```
// sb-reverse.jsl
// StringBuffer.reverse example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("Hello world!");

        // Reverse characters and display the buffer:
        System.out.println("Reversed Hello world!: " + s.reverse());
    }
}

/*
Output:
Reversed Hello world!: !dlrow olleH
*/
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.setCharAt Method

Changes a character at a specified index in a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized void setCharAt(
    int index,
    char ch);
```

Parameters

index

The index of the character to be changed.

ch

The new character.

Example

```
// sb-setChAt.jsl
// StringBuffer.setCharAt example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("Hello world!");

        // Change w to W:
        s.setCharAt(6,'W');

        // Display the buffer:
        System.out.println(s);
    }
}

/*
Output:
Hello World!
*/
```

Remarks

Throws StringIndexOutOfBoundsException if index is negative or greater than or equal to the current length of the buffer.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.setLength Method

Changes the length of a StringBuffer object to a specified length.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized void setLength(
    int newLength);
```

Parameters

newLength

The new length of the buffer.

Example

```
// sb-setLent.jsl
// StringBuffer.setLength example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("Hello world!");

        // Change the length to 5 characters:
        s.setLength(5);

        // Display the buffer:
        System.out.println(s);
    }
}

/*
Output:
Hello
*/
```

Remarks

Throws StringIndexOutOfBoundsException if newLength is negative.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.substring Method

Overload List

Name	Description
StringBuffer.substring (int)	Returns a substring that starts at a specific index in a StringBuffer object.
StringBuffer.substring (int, int)	Returns the substring that begins at the specified starting index and extends up to, but not including, the character at the ending index.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.substring Method (Int32)

Returns a substring that starts at a specific index in a StringBuffer object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String substring(
    int start);
```

Parameters

start

The starting index of the substring.

Return Value

The substring that starts at start and extends to the end of the string.

Example

```
// sb-substring1.jsl
// StringBuffer.substring example

public class MyClass
{
    public static void main(String[] args)
    {
      // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("Hello world!");

        // Display the substring:
        System.out.println("In \"" + s +
        "\", the substring starting at the 6th character is: "
        + s.substring(6));
    }
}

/*
Output:
In "Hello world!", the substring starting at the 6th character is: world!
*/
```

Remarks

Throws StringIndexOutOfBoundsException if start is negative or greater than the length of StringBuffer.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.substring Method (Int32, Int32)

Returns the substring that begins at the specified starting index and extends up to, but not including, the character at the ending index.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.String substring(
    int start,
    int end);
```

Parameters

start

The starting index (inclusive).

end

The ending index (exclusive).

Return Value

The string specified by the start and end indexes.

Example

```
// sb-substring2.jsl
// StringBuffer.subString example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("Hello world!");

        // Display the substring:
        System.out.println("In \"" + s +
        "\", the substring starting from 6th character up to, " +
        "but not including, the 11th character is: "
        + s.substring(6,11));
    }
}

/*
Output:
In "Hello world!", the substring starting from 6th character up to, but not including, the
11th character is: world
*/
```

Remarks

Throws StringIndexOutOfBoundsException if start or end is negative, if end is greater than the length of StringBuffer, or if start is greater than end.

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringBuffer.toString Method

Converts a StringBuffer object to a String object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The String object equivalent to the value stored in the StringBuffer object.

Example

```
// sb-toStr.jsl
// StringBuffer.toString example

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a StringBuffer object:
        StringBuffer s = new StringBuffer("Hello there");

        // Append a char to the buffer:
        s.append('!');

       // Convert to a String object and display it:
       System.out.println(s.toString());
    }
}

/*
Output:
Hello there!
*/
```

See Also

Reference

StringBuffer Class

Concepts

StringBuffer Members

java.lang Package

StringIndexOutOfBoundsException Class

An exception that is thrown by a string method when an index is greater than the size of the string

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.StringIndexOutOfBoundsException
    extends java.lang.IndexOutOfBoundsException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.IndexOutOfBoundsException

java.lang.StringIndexOutOfBoundsException

See Also

Concepts

StringIndexOutOfBoundsException Members

java.lang Package

StringIndexOutOfBoundsException Members

An exception that is thrown by a string method when an index is greater than the size of the string

The following tables list the members exposed by the StringIndexOutOfBoundsException type.

Public Constructors

Name	Description
StringIndexOutOfBoundsException	Overloaded. Constructs a StringIndexOutOfBoundsException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringIndexOutOfBoundsException Class

Concepts

java.lang Package

StringIndexOutOfBoundsException Constructor

Constructs a StringIndexOutOfBoundsException object.

Overload List

Name	Description
StringIndexOutOfBoundsException ()	Constructs a StringIndexOutOfBoundsException object without a specified text message.
StringIndexOutOfBoundsException (int)	Constructs a StringIndexOutOfBoundsException object with a specified element index.
StringIndexOutOfBoundsException (String)	Constructs a StringIndexOutOfBoundsException object with a specified text message.
StringIndexOutOfBoundsException (SerializationInfo, StreamingContext)	Constructs a StringIndexOutOfBoundsException object during serialization.
StringIndexOutOfBoundsException (String, Exception)	Constructs a StringIndexOutOfBoundsException object with a specific text message and inner exception.

See Also

Reference

StringIndexOutOfBoundsException Class

Concepts

StringIndexOutOfBoundsException Members

java.lang Package

StringIndexOutOfBoundsException Constructor ()

Constructs a StringIndexOutOfBoundsException object without a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringIndexOutOfBoundsException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

StringIndexOutOfBoundsException Class

Concepts

StringIndexOutOfBoundsException Members

java.lang Package

StringIndexOutOfBoundsException Constructor (Int32)

Constructs a StringIndexOutOfBoundsException object with a specified element index.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringIndexOutOfBoundsException(
    int i);
```

Parameters

i

The index of the string array element that exceeded the limit.

See Also
Reference
StringIndexOutOfBoundsException Class
Concepts
StringIndexOutOfBoundsException Members
java.lang Package

StringIndexOutOfBoundsException Constructor (String)

Constructs a StringIndexOutOfBoundsException object with a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringIndexOutOfBoundsException(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

StringIndexOutOfBoundsException Class

Concepts

StringIndexOutOfBoundsException Members

java.lang Package

StringIndexOutOfBoundsException Constructor (SerializationInfo, StreamingContext)

Constructs a StringIndexOutOfBoundsException object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.StringIndexOutOfBoundsException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also
Reference
StringIndexOutOfBoundsException Class
Concepts
StringIndexOutOfBoundsException Members
java.lang Package

StringIndexOutOfBoundsException Constructor (String, Exception)

Constructs a StringIndexOutOfBoundsException object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringIndexOutOfBoundsException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to StringIndexOutOfBoundsException.

See Also
Reference
StringIndexOutOfBoundsException Class
Concepts
StringIndexOutOfBoundsException Members
java.lang Package

StringIndexOutOfBoundsException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringIndexOutOfBoundsException Class

Concepts

java.lang Package

StringIndexOutOfBoundsException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

StringIndexOutOfBoundsException Class

Concepts

java.lang Package

System Class

Contains useful fields such as the standard input, standard output, and standard error streams. It also contains methods to access properties, load files and libraries, and copy arrays.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.System
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.lang.System

See Also

Concepts

System Members

java.lang Package

System Members

Contains useful fields such as the standard input, standard output, and standard error streams. It also contains methods to access properties, load files and libraries, and copy arrays.

The following tables list the members exposed by the System type.

Public Fields

Name	Description
err	The standard error stream.
in	The standard input stream.
out	The standard output stream.

Public Methods

Name	Description
arraycopy	Copies a specified portion of a source array, starting from a specified position, to a destination array, starting at a specified position.
currentTimeMillis	Retrieves the current time in milliseconds
equals	(inherited from Object)
exit	Terminates the current application and exits with a specified status code.
gc	Invokes the garbage collector.
getenv	This method is replaced by getProperty.
hashCode	(inherited from Object)
getProperties	Retrieves the current system properties.
getProperty	Overloaded.
getSecurityManager	Returns the security manager.
getClass	(inherited from Object)
identityHashCode	Retrieves the hash code for a specified object.
load	Loads a specified file.
loadLibrary	Loads a specified library.
clone	(inherited from Object)
runFinalization	Runs the finalization methods on objects waiting to be finalized.
runfinalizersOnExit	Runs the garbage collection when the application exits.

setErr	Sets the standard error stream to a custom error stream..
setIn	Sets the standard input stream to a custom input stream.
setOut	Sets the standard output stream to a custom output stream.
setProperties	Sets the system properties using a specified parameter.
setSecurityManager	Sets the security manager. Obsolete.
toStrin	Overridden. Returns the string representation of the current object.

See Also

Reference

System Class

Concepts

java.lang Package

System Fields

Public Fields

Name	Description
err	The standard error stream.
in	The standard input stream.
out	The standard output stream.

See Also

Reference

System Class

Concepts

java.lang Package

System.err Field

The standard error stream.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.io.PrintStream err;
```

Example

```
// sys-err.jsl
// System.err example

public class MyClass
{
    public static void main(String[] args)
    {
        // Write to the error stream:
        System.err.print("Hello world!");
    }
}

/*
Output:
Hello world!
*/
```

Remarks

Typically, this stream is used for displaying error massages, especially when the standard output stream is redirected to a file and cannot be monitored.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.in Field

The standard input stream.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.io.InputStream in;
```

Example

```
// sys-in.jsl
// System.in example

public class MyClass
{
    public static void main(String[] args) throws java.io.IOException
    {
         char c = (char)System.in.read();
         System.out.print("You typed: " + c);
    }
}

/*
Input:
E
Output:
You typed: E
*/
```

Remarks

Typically, the input stream corresponds to the keyboard.

The java.io.IOException is expected with input operations. Use throws or try-catch statements to avoid exceptions.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.out Field

The standard output stream.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.io.PrintStream out;
```

Example

```
// sys-out.jsl
// System.out example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print("Hello world!");
    }
}

/*
Output:
Hello world!
*/
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System Methods

Public Methods

Name	Description
arraycopy	Copies a specified portion of a source array, starting from a specified position, to a destination array, starting at a specified position.
currentTimeMillis	Retrieves the current time in milliseconds
equals	(inherited from Object)
exit	Terminates the current application and exits with a specified status code.
gc	Invokes the garbage collector.
getenv	This method is replaced by getProperty.
hashCode	(inherited from Object)
getProperties	Retrieves the current system properties.
getProperty	Overloaded.
getSecurityManager	Returns the security manager.
getClass	(inherited from Object)
identityHashCode	Retrieves the hash code for a specified object.
load	Loads a specified file.
loadLibrary	Loads a specified library.
runFinalization	Runs the finalization methods on objects waiting to be finalized.
runfinalizersOnExit	Runs the garbage collection when the application exits.
setErr	Sets the standard error stream to a custom error stream..
setIn	Sets the standard input stream to a custom input stream.
setOut	Sets the standard output stream to a custom output stream.
setProperties	Sets the system properties using a specified parameter.
setSecurityManager	Sets the security manager. Obsolete.
toString	Overridden. Returns the string representation of the current object.

See Also

Reference

System Class

Concepts
java.lang Package

System.arraycopy Method

Copies a specified portion of a source array, starting from a specified position, to a destination array, starting at a specified position.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void arraycopy(
    java.lang.Object src,
    int src_position,
    java.lang.Object dst,
    int dst_position,
    int length);
```

Parameters

src

The source array object.

src_position

The position where the copying starts in the source array.

dst

The destination array object.

dst_position

The position where the copying starts in the destination array.

length

The length of the array's portion to be copied.

Example

```
// sys-arr1.jsl
// System.arraycopy example

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare two arrays:
        int[] arr1 = {1,2,3,4,5,6};
        int[] arr2 = {0,2,4,6,8,10};

        // Copy two elements from arr1 starting from the second element
        // into arr2 starting at the fourth element:
        System.arraycopy(
                        arr1,
                        1,       // Second element.
                        arr2,
                        3,       // Fourth element.
                        2        // Length = 2.
                        );

        // Display the new array (arr2):
        for (int i=0; i<arr2.length; i++)
            System.out.println("Element #" + i + " = " + arr2[i]);
    }
}

/*
```

```
Output:
Element #0 = 0
Element #1 = 2
Element #2 = 4
Element #3 = 2
Element #4 = 3
Element #5 = 10
*/
```

Remarks

Throws java.lang.ArrayIndexOutOfBoundsException if any of the following conditions is true:

dst or src is an empty array.

src_position is negative

dst_position is negative.

length is negative.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.currentTimeMillis Method

Retrieves the current time in milliseconds

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static long currentTimeMillis();
```

Return Value

A long value that represents the current time in milliseconds.

Example

```
// sys-crntime.jsl
// System.currentTimeMillis example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.print(System.currentTimeMillis());
    }
}

/*
1094596466359
*/
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.exit Method

Terminates the current application and exits with a specified status code.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void exit(
    int status);
```

Parameters

status

An int indicating the status code.

Remarks

Usually, the status code 0 indicates normal exit.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.gc Method

Invokes the garbage collector.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void gc();
```

Example

```
// sys-gc1.jsl
// System.gc example

public class MyClass
{
    public static void main(String[] args)
    {
        System.gc();
        System.out.println("Cleanup done!");
    }
}

/*
Output:
Cleanup done!
*/
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.getenv Method

This method is replaced by getProperty.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String getenv(
    java.lang.String name);
```

Parameters

name

The variable name.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.getProperties Method

Retrieves the current system properties.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Properties getProperties();
```

Return Value

The current system properties.

Example

```
// sys-getProps1.jsl
// System.getProperties example

public class MyClass
{
    public static void main(String[] args)
    {
         System.out.println(System.getProperties());
    }
}

/*
Output:
{com.ms.locale.variant=, user.region=US, file.separator=\, file.encoding=Cp1252,
 browser=ActiveX Scripting Host, http.agent=Mozilla/4.0 (compatible; MSIE 6.0; W
in32), user.name=samabo, user.language=en, jbcu.windowWarning=Warning - J# Brows
er Control window, line.separator=
, awt.toolkit=com.ms.vjsharp.windowing.win32.Win32Toolkit, os.arch=x86, os.name=
Windows XP, com.ms.windir=C:\WINDOWS, os.version=5.1, com.ms.sysdir=C:\WINDOWS\S
ystem32, com.ms.applet.enable.serversockets=false, http.ProxyPort=80, http.Proxy
Host=//itgproxy, user.home=C:\WINDOWS, path.separator=;, user.dir=C:\jcompiler\L
ang\System, user.timezone=PST}
*/
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.getProperty Method

Overload List

Name	Description
System.getProperty (String)	Returns the system property associated with a specified key.
System.getProperty (String, String)	Returns the system property associated with a specified key.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.getProperty Method (String)

Returns the system property associated with a specified key.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String getProperty(
    java.lang.String key);
```

Parameters

key

The key that corresponds to the property.

Return Value

The system property specified by key.

Example

```
// sys-getProp1.jsl
// System.getProperty example

public class MyClass
{
    public static void main(String[] args)
    {
         String s  = System.getProperty("os.version");

         // Notice that the output of the following statement can be
         // different for different machines:
         System.out.print("The OS version for this machine is: " + s);
    }
}

/*
Output:
The OS version for this machine is: 5.1
*/
```

Remarks

Throws java.lang.IllegalArgumentException if the key is null.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.getProperty Method (String, String)

Returns the system property associated with a specified key.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String getProperty(
    java.lang.String key,
    java.lang.String def);
```

Parameters

key

The key string.

def

The default name of the property.

Return Value

The system property associated with key or def if there is no property associated with the specified key.

Example

```
// sys-getProp2.jsl
// System.getProperty example

public class MyClass
{
    public static void main(String[] args)
    {
         String s1  = System.getProperty("os.name","My default");
         // Notice that the property key is misspelled here:
         String s2  = System.getProperty("os-name","My default");

         // Notice that the output of the following statement can be
         // different for different operating systems:
         System.out.println("The OS name is: " + s1);

         // The following statement displays the default property name:
         System.out.println("The OS name is: " + s2);
    }
}

/*
Output:
The OS name is: Windows XP
The OS name is: My default
*/
```

Remarks

Throws java.lang.IllegalArgumentException if the key is null.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.getSecurityManager Method

Returns the security manager.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.SecurityManager getSecurityManager();
```

Return Value

The security manager if it has been set; null otherwise.

Example

```
// sys-getsec1.jsl
// System.getSecurityManager example

public class MyClass
{
    public static void main(String[] args)
    {
        System.out.println(System.getSecurityManager());
    }
}

/*
Output:
null
*/
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.identityHashCode Method

Retrieves the hash code for a specified object.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static int identityHashCode(
    java.lang.Object x);
```

Parameters

x

The object for which the hash code is retrieved.

Return Value

The hash code for the specified object.

Example

```
// sys-idHashCode1.jsl
// System.identityHashCode example

public class MyClass
{
    public static void main(String[] args)
    {
        int hc = System.identityHashCode(new Double(1.23));
        System.out.println(hc);
    }
}

/*
Output:
18643596
*/
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.load Method

Loads a specified file.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void load(
    java.lang.String filename);
```

Parameters

filename

The file to be loaded.

Example

```
// sys-load1.jsl
// System.load example

public class MyClass
{
    public static void main(String[] args)
    {
        System.load("C:\\Lang\\System\\sys-gc1.exe");
        System.out.println("File loaded.");
    }
}

/*
Output:
File loaded.
*/
```

Remarks

The full path of the file to be loaded must me specified.

Throws java.lang.UnsatisfiedLinkError if the file does not exist.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.loadLibrary Method

Loads a specified library.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void loadLibrary(
    java.lang.String libname);
```

Parameters

libname

The library to be loaded.

Example

```
// sys-load2.jsl
// System.load example

public class MyClass
{
    public static void main(String[] args)
    {
        System.load("C:\\assembly\\vjslib.dll");
        System.out.println("Library loaded.");
    }
}

/*
Output:
Library loaded.
*/
```

Remarks

The full path of the file to be loaded must me specified.

Throws java.lang.UnsatisfiedLinkError if the library file does not exist.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.runFinalization Method

Runs the finalization methods on objects waiting to be finalized.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void runFinalization();
```

Example

```
// sys-runF1.jsl
// System.runFinalization example

public class MyClass
{
    public static void main(String[] args)
    {
        System.runFinalization();
        System.out.println("Cleanup done!");
    }
}

/*
Output:
Cleanup done!
*/
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.runfinalizersOnExit Method

Runs the garbage collection when the application exits.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void runFinalizersOnExit(
    boolean value);
```

Parameters

value

A value to indicate the

Example

```
// sys-runF2.jsl
// System.runFinalizersOnExit example

public class MyClass
{
    public static void main(String[] args)
    {
        System.runFinalizersOnExit(true);
        System.out.println("Cleanup done!");
    }
}

/*
Output:
Cleanup done!
*/
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.setErr Method

Sets the standard error stream to a custom error stream..

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final void setErr(
    java.io.PrintStream errStream);
```

Parameters

errStream

The custom error stream.

Example

```
// sys-seEr1.jsl
// System.setErr example

import java.io.*;

public class MyClass
{
    public static void main(String[] args) throws java.io.IOException
    {
        // Create a file:
        System.setErr(new PrintStream(
                 new FileOutputStream("MyErrorFile.txt")));
        // Redirect the output to the file:
        System.err.println("Hello to custom error stream!");
    }
}
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.setIn Method

Sets the standard input stream to a custom input stream.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final void setIn(
    java.io.InputStream inStream);
```

Parameters

inStream

The custom input stream.

Example

In this example, you create a text file (InputFile.txt) that contains some numbers and the character 'E' at the end. Then you assign this file to the input stream, read the first char, and keep reading from the file until the char 'E' is encountered.

```
// System.setIn example

import java.io.*;

public class MyClass
{
    public static void main(String[] args) throws java.io.IOException
    {
        // Assign an existing input file to input stream:
        System.setIn(new FileInputStream("InputFile.txt"));

        // Read the first char in the file:
        int oneByteFromInFile = (char)System.in.read();

        // Read and display chars until you encounter 'E':
        while (oneByteFromInFile != 'E')
        {
            System.out.print((char)oneByteFromInFile + " ");
            oneByteFromInFile = (char)System.in.read();
        }

        // Display a success message:
        System.out.println("\nAll chars have been read!");
    }
}

/*
Output:
1 2 3 4 5 6 7 8 9
All chars have been read!
*/
```

Remarks

The method is successful if there are no security restrictions on re-assigning the input stream.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.setOut Method

Sets the standard output stream to a custom output stream.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final void setOut(
    java.io.PrintStream outStream);
```

Parameters

outStream

The custom output stream.

Example

In this example, you create the file MyOutputFile.txt and redirect the output to this file. After you run the program you can examine the contents of the file.

```
// sys-seto1.jsl
// System.setout example

import java.io.*;

public class MyClass
{
    public static void main(String[] args) throws java.io.IOException
    {
        // Create a file:
        System.setOut(new PrintStream(
                 new FileOutputStream("MyOutputFile.txt")));
        // Redirect the output to the file:
        System.out.println("Hello to custom output stream!");
    }
}
```

Remarks

The method is successful if there are no security restrictions on re-assigning the output stream.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.setProperties Method

Sets the system properties using a specified parameter.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void setProperties(
    java.util.Properties props);
```

Parameters

props

The new system properties.

Example

```
// sys-setprop1.jsl
// System.setProperties example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        Properties p = new Properties();

        // Set system properties using the newly allocated variable:
        System.setProperties(p);

        // The following statement is equivalent to
        // System.out.println(p). Both print an empty
        // set of properties:
        System.out.println(System.getProperties());
    }
}

/*
Output:
{}
*/
```

See Also

Reference

System Class

Concepts

System Members

java.lang Package

System.setSecurityManager Method

NOTE: This Method is now obsolete.

Sets the security manager.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static void setSecurityManager(
    java.lang.SecurityManager s);
```

Parameters

s

The name of the security manager.

See Also

Reference

System Class

Concepts

System Members

java.lang Package

Throwable Class

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.Throwable
    extends java.lang.Object
    implements java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

com.ms.vjsharp.lang.ThrowableWrapper

java.lang.Error

java.lang.Exception

See Also

Concepts

Throwable Members

java.lang Package

Throwable Members

The following tables list the members exposed by the Throwable type.

Public Constructors

Name	Description
Throwable	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	
equals	(inherited from Object)
fillInStackTrace	
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	
getMessage	
GetObjectData	Overridden.
getClass	(inherited from Object)
clone	
printStackTrace	Overloaded.

toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Throwable Class

Concepts

java.lang Package

Throwable Constructor

Overload List

Name	Description
Throwable ()	
Throwable (String)	
Throwable (SerializationInfo, StreamingContext)	
Throwable (String, Exception)	

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable Constructor ()

Initializes a new instance of the Throwable Class .

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Throwable();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable Constructor (String)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Throwable(
    java.lang.String message);
```

Parameters

message

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable Constructor (SerializationInfo, StreamingContext)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Throwable(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable Constructor (String, Exception)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Throwable(
    java.lang.String message,
    System.Exception innerEx);
```

Parameters

message

innerEx

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable Methods

Public Methods

Name	Description
<exceptFilter>	
equals	(inherited from Object)
fillInStackTrace	
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	
getMessage	
GetObjectData	Overridden.
getClass	(inherited from Object)
clone	
printStackTrace	Overloaded.
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Throwable Class

Concepts

java.lang Package

Throwable.fillInStackTrace Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Throwable fillInStackTrace();
```

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable.getLocalizedMessage Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getLocalizedMessage();
```

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable.getMessage Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getMessage();
```

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable.GetObjectData Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable.printStackTrace Method

Overload List

Name	Description
Throwable.printStackTrace ()	
Throwable.printStackTrace (PrintStream)	
Throwable.printStackTrace (PrintWriter)	

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable.printStackTrace Method ()

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public void printStackTrace();
```

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable.printStackTrace Method (PrintStream)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public void printStackTrace(
    java.io.PrintStream s);
```

Parameters

s

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable.printStackTrace Method (PrintWriter)

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public void printStackTrace(
    java.io.PrintWriter wr);
```

Parameters

wr

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable.toString Method

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

Throwable Class

Concepts

Throwable Members

java.lang Package

Throwable Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

Throwable Class

Concepts

java.lang Package

UnknownError Class

An error that is issued when an unknown error occurs.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.UnknownError
    extends java.lang.VirtualMachineError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.VirtualMachineError

java.lang.UnknownError

See Also

Concepts

UnknownError Members

java.lang Package

UnknownError Members

An error that is issued when an unknown error occurs.

The following tables list the members exposed by the UnknownError type.

Public Constructors

Name	Description
UnknownError	Overloaded. Constructs a UnknownError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UnknownError Class

Concepts

java.lang Package

UnknownError Constructor

Constructs a UnknownError object.

Overload List

Name	Description
UnknownError ()	Constructs a UnknownError object without a specified text message.
UnknownError (String)	Constructs a UnknownError object with a specified text message.
UnknownError (SerializationInfo, StreamingContext)	Constructs a UnknownError object during serialization.
UnknownError (String, Exception)	a UnknownError object with a specific text message and inner exception.

See Also

Reference

UnknownError Class

Concepts

UnknownError Members

java.lang Package

UnknownError Constructor ()

Constructs a UnknownError object without a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.UnknownError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

UnknownError Class

Concepts

UnknownError Members

java.lang Package

UnknownError Constructor (String)

Constructs a UnknownError object with a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.UnknownError(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also
Reference
UnknownError Class
Concepts
UnknownError Members
java.lang Package

UnknownError Constructor (SerializationInfo, StreamingContext)

Constructs a UnknownError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.UnknownError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also
Reference
UnknownError Class
Concepts
UnknownError Members
java.lang Package

UnknownError Constructor (String, Exception)

a UnknownError object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.UnknownError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to UnknownError.

See Also

Reference

UnknownError Class

Concepts

UnknownError Members

java.lang Package

UnknownError Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UnknownError Class

Concepts

java.lang Package

UnknownError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

UnknownError Class

Concepts

java.lang Package

UnsatisfiedLinkError Class

An error that is issued when the execution engine cannot find the appropriate definition for a declared method.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.UnsatisfiedLinkError
    extends java.lang.LinkageError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.UnsatisfiedLinkError

See Also

Concepts

UnsatisfiedLinkError Members

java.lang Package

UnsatisfiedLinkError Members

An error that is issued when the execution engine cannot find the appropriate definition for a declared method.

The following tables list the members exposed by the UnsatisfiedLinkError type.

Public Constructors

Name	Description
UnsatisfiedLinkError	Overloaded. Constructs an UnsatisfiedLinkError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an error object to an UnsatisfiedLinkError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UnsatisfiedLinkError Class

Concepts

java.lang Package

UnsatisfiedLinkError Constructor

Constructs an UnsatisfiedLinkError object.

Overload List

Name	Description
UnsatisfiedLinkError ()	Constructs an UnsatisfiedLinkError object without a specific text message.
UnsatisfiedLinkError (String)	Constructs an UnsatisfiedLinkError object with a specific text message.
UnsatisfiedLinkError (SerializationInfo, StreamingContext)	Constructs an UnsatisfiedLinkError object during serialization.
UnsatisfiedLinkError (String, Exception)	Constructs an UnsatisfiedLinkError object with a specific text message and inner exception.

See Also

Reference

UnsatisfiedLinkError Class

Concepts

UnsatisfiedLinkError Members

java.lang Package

UnsatisfiedLinkError Constructor ()

Constructs an UnsatisfiedLinkError object without a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.UnsatisfiedLinkError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

UnsatisfiedLinkError Class

Concepts

UnsatisfiedLinkError Members

java.lang Package

UnsatisfiedLinkError Constructor (String)

Constructs an UnsatisfiedLinkError object with a specific text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.UnsatisfiedLinkError(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

UnsatisfiedLinkError Class

Concepts

UnsatisfiedLinkError Members

java.lang Package

UnsatisfiedLinkError Constructor (SerializationInfo, StreamingContext)

Constructs an UnsatisfiedLinkError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.UnsatisfiedLinkError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object

context

The source or destination for the serialization.

See Also

Reference

UnsatisfiedLinkError Class

Concepts

UnsatisfiedLinkError Members

java.lang Package

UnsatisfiedLinkError Constructor (String, Exception)

Constructs an UnsatisfiedLinkError object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.UnsatisfiedLinkError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to UnsatisfiedLinkError.

See Also

Reference

UnsatisfiedLinkError Class

Concepts

UnsatisfiedLinkError Members

java.lang Package

UnsatisfiedLinkError Methods

Public Methods

Name	Description
<exceptFilter>	Maps an error object to an UnsatisfiedLinkError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UnsatisfiedLinkError Class

Concepts

java.lang Package

UnsatisfiedLinkError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

UnsatisfiedLinkError Class

Concepts

java.lang Package

UnsupportedOperationException Class

An exception that is thrown when an application tries to perform an unsupported operation.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.UnsupportedOperationException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.lang.UnsupportedOperationException

See Also

Concepts

UnsupportedOperationException Members

java.lang Package

UnsupportedOperationException Members

An exception that is thrown when an application tries to perform an unsupported operation.

The following tables list the members exposed by the UnsupportedOperationException type.

Public Constructors

Name	Description
UnsupportedOperationException	Overloaded. Constructs a UnsupportedOperationException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UnsupportedOperationException Class

Concepts

java.lang Package

UnsupportedOperationException Constructor

Constructs a UnsupportedOperationException object.

Overload List

Name	Description
UnsupportedOperationException ()	Constructs a UnsupportedOperationException object without a specified text message.
UnsupportedOperationException (String)	Constructs a UnsupportedOperationException object with a specified text message.
UnsupportedOperationException (SerializationInfo, StreamingContext)	Constructs a UnsupportedOperationException object during serialization.
UnsupportedOperationException (String, Exception)	Constructs a UnsupportedOperationException object with a specific text message and inner exception.

See Also

Reference

UnsupportedOperationException Class

Concepts

UnsupportedOperationException Members

java.lang Package

UnsupportedOperationException Constructor ()

Constructs a UnsupportedOperationException object without a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.UnsupportedOperationException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

UnsupportedOperationException Class

Concepts

UnsupportedOperationException Members

java.lang Package

UnsupportedOperationException Constructor (String)

Constructs a UnsupportedOperationException object with a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.UnsupportedOperationException(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also
Reference
UnsupportedOperationException Class
Concepts
UnsupportedOperationException Members
java.lang Package

UnsupportedOperationException Constructor (SerializationInfo, StreamingContext)

Constructs a UnsupportedOperationException object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.UnsupportedOperationException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

UnsupportedOperationException Class

Concepts

UnsupportedOperationException Members

java.lang Package

UnsupportedOperationException Constructor (String, Exception)

Constructs a UnsupportedOperationException object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.UnsupportedOperationException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to UnsupportedOperationException.

See Also
Reference
UnsupportedOperationException Class
Concepts
UnsupportedOperationException Members
java.lang Package

UnsupportedOperationException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

UnsupportedOperationException Class

Concepts

java.lang Package

UnsupportedOperationException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

UnsupportedOperationException Class

Concepts

java.lang Package

VerifyError Class

An error issued when the execution engine cannot verify a class code because it contains issues related to class format semantics.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.VerifyError
    extends java.lang.LinkageError
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.LinkageError

java.lang.VerifyError

See Also

Concepts

VerifyError Members

java.lang Package

VerifyError Members

An error issued when the execution engine cannot verify a class code because it contains issues related to class format semantics.

The following tables list the members exposed by the VerifyError type.

Public Constructors

Name	Description
VerifyError	Overloaded. Constructs a VerifyError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a VerifyError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

VerifyError Class

Concepts

java.lang Package

VerifyError Constructor

Constructs a VerifyError object.

Overload List

Name	Description
VerifyError ()	Constructs a VerifyError object without a specified text message.
VerifyError (String)	Constructs a VerifyError object with a specified text message.
VerifyError (SerializationInfo, StreamingContext)	Constructs VerifyError object during serialization.
VerifyError (String, Exception)	Constructs a VerifyError object with a specific text message and inner exception.

See Also

Reference

VerifyError Class

Concepts

VerifyError Members

java.lang Package

VerifyError Constructor ()

Constructs a VerifyError object without a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.VerifyError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

VerifyError Class

Concepts

VerifyError Members

java.lang Package

VerifyError Constructor (String)

Constructs a VerifyError object with a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.VerifyError(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

VerifyError Class

Concepts

VerifyError Members

java.lang Package

VerifyError Constructor (SerializationInfo, StreamingContext)

Constructs VerifyError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.VerifyError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

VerifyError Class

Concepts

VerifyError Members

java.lang Package

VerifyError Constructor (String, Exception)

Constructs a VerifyError object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.VerifyError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to VerifyError.

See Also

Reference

VerifyError Class

Concepts

VerifyError Members

java.lang Package

VerifyError Methods

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a VerifyError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

VerifyError Class

Concepts

java.lang Package

VerifyError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

VerifyError Class

Concepts

java.lang Package

VirtualMachineError Class

An error issued to indicate that there is a problem with the execution engine.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.VirtualMachineError
    extends java.lang.Error
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Error

java.lang.VirtualMachineError

java.lang.InternalError

java.lang.OutOfMemoryError

java.lang.StackOverflowError

java.lang.UnknownError

See Also

Concepts

VirtualMachineError Members

java.lang Package

VirtualMachineError Members

An error issued to indicate that there is a problem with the execution engine.

The following tables list the members exposed by the VirtualMachineError type.

Public Constructors

Name	Description
VirtualMachineError	Overloaded. Constructs a VirtualMachineError object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a VirtualMachineError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)

printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

VirtualMachineError Class

Concepts

java.lang Package

VirtualMachineError Constructor

Constructs a VirtualMachineError object.

Overload List

Name	Description
VirtualMachineError ()	Constructs a VirtualMachineError object without a specified text message.
VirtualMachineError (String)	Constructs a VirtualMachineError object with a specified text message.
VirtualMachineError (SerializationInfo, StreamingContext)	Constructs a VirtualMachineError object during serialization.
VirtualMachineError (String, Exception)	Constructs a VirtualMachineError object with a specific text message and inner exception.

See Also

Reference

VirtualMachineError Class

Concepts

VirtualMachineError Members

java.lang Package

VirtualMachineError Constructor ()

Constructs a VirtualMachineError object without a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.VirtualMachineError();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

VirtualMachineError Class

Concepts

VirtualMachineError Members

java.lang Package

VirtualMachineError Constructor (String)

Constructs a VirtualMachineError object with a specified text message.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.VirtualMachineError(
    java.lang.String s);
```

Parameters

s

The string that contains the message text.

See Also

Reference

VirtualMachineError Class

Concepts

VirtualMachineError Members

java.lang Package

VirtualMachineError Constructor (SerializationInfo, StreamingContext)

Constructs a VirtualMachineError object during serialization.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.VirtualMachineError(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

VirtualMachineError Class

Concepts

VirtualMachineError Members

java.lang Package

VirtualMachineError Constructor (String, Exception)

Constructs a VirtualMachineError object with a specific text message and inner exception.

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public java.lang.VirtualMachineError(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to VirtualMachineError.

See Also

Reference

VirtualMachineError Class

Concepts

VirtualMachineError Members

java.lang Package

VirtualMachineError Methods

Public Methods

Name	Description
<exceptFilter>	Maps an error object to a VirtualMachineError object.
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

VirtualMachineError Class

Concepts

java.lang Package

VirtualMachineError Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

VirtualMachineError Class

Concepts

java.lang Package

Void Class

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.Void
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.lang.Void

See Also

Concepts

Void Members

java.lang Package

Void Members

The following tables list the members exposed by the Void type.

Public Fields

Name	Description
TYPE	

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
toString	Overridden.

See Also

Reference

Void Class

Concepts

java.lang Package

Void Fields

Public Fields

Name	Description
TYPE	

See Also

Reference

Void Class

Concepts

java.lang Package

Void.TYPE Field

Package: java.lang

Assembly: vjslib (in vjslib.dll)

```
public static final java.lang.Class TYPE;
```

See Also

Reference

Void Class

Concepts

Void Members

java.lang Package

Void Methods

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
toString	Overridden.

See Also

Reference

Void Class

Concepts

java.lang Package

java.lang.Class

Represents class types and interface types. Using this class, you can determine the constructors, methods, fields, and other information of a type at runtime.

J#

```
public final class Class implements Serializable
```

Example

```
// class_overview.jsl

import java.lang.reflect.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with
        // "java.lang.String".
        try
        {
            Class stringClass = Class.forName("java.lang.String");
            System.out.println("Class found: " +
                stringClass.getName());

            // Get the Class object associated with "Program".
            Class thisClass = Class.forName("Program");

            // Get the name of this class.
            String name = thisClass.getName();
            System.out.println("\nThis class name: " + name);

            // Get all the public Class objects associated with the
            // Program class.
            Class[] classes = thisClass.getClasses();
            System.out.println();
            for (int i = 0; i < classes.length; i++)
            {
                System.out.println("Class found: " +
                    classes[i].getName());
            }

            // Get all the Class objects associated with the
            // Program class.
            classes = thisClass.getDeclaredClasses();
            System.out.println();
            for (int i = 0; i < classes.length; i++)
            {
                System.out.println("Class found: " +
                    classes[i].getName());
            }

            // Get all the public constructors for ths class.
            Constructor[] ctors = thisClass.getConstructors();
            System.out.println();
            for (int i = 0; i < ctors.length; i++)
            {
                System.out.println("Public constructor found: " +
                    ctors[i].toString());
            }

            // Get all the constructors for ths class.
            ctors = thisClass.getDeclaredConstructors();
```

```
            System.out.println();
            for (int i = 0; i < ctors.length; i++)
            {
                System.out.println("Constructor found: " +
                    ctors[i].toString());
            }

            // Get all the public fields associated with this class.
            Field[] fields = thisClass.getFields();
            System.out.println();
            for (int i = 0; i < fields.length; i++)
            {
                System.out.println("Public field found: " +
                    fields[i].toString());
            }

            // Get all the fields associated with this class.
            fields = thisClass.getDeclaredFields();
            System.out.println();
            for (int i = 0; i < fields.length; i++)
            {
                System.out.println("Field found: " +
                    fields[i].toString());
            }

            // Get the public methods associated with this class.
            Method[] methods = thisClass.getMethods();
            System.out.println();
            for (int i = 0; i < methods.length; i++)
            {
                System.out.println("Public method found: " +
                    methods[i].toString());
            }

            // Get all the methods associated with this class.
            methods = thisClass.getDeclaredMethods();
            System.out.println();
            for (int i = 0; i < methods.length; i++)
            {
                System.out.println("Method found: " +
                    methods[i].toString());
            }

            // Get the interfaces associated with this class.
            Class[] interfaces = thisClass.getInterfaces();
            System.out.println();
            for (int i = 0; i < interfaces.length; i++)
            {
                System.out.println("Interface found: " +
                    interfaces[i].toString());
            }

            // Get the modifiers for this class.
            int i = thisClass.getModifiers();
            String mods = Modifier.toString(i);
            System.out.println("\nClass modifiers: " + mods);

            // Get the super class of this class.
            Class superClass = thisClass.getSuperclass();
            System.out.println("\nSuper class name: " +
                superClass.getName());
        }
        catch (ClassNotFoundException ex)
        {
            System.out.println(ex.toString());
        }
    }
```

```
    public class InnerPublicClass
    {
        public InnerPublicClass()
        {
        }
    }

    private class InnerPrivateClass
    {
        public InnerPrivateClass()
        {
        }
    }
}

/*
Output:
Class found: java.lang.String

This class name: Program

Class found: Program$InnerPublicClass

Class found: Program$InnerPublicClass
Class found: Program$InnerPrivateClass

Public constructor found: public Program()

Constructor found: public Program()



Public method found: public final java.lang.Class java.lang.Object.getClass()
Public method found: public final void java.lang.Object.notify()
Public method found: public final void java.lang.Object.notifyAll()
Public method found: public final void java.lang.Object.wait() throws java.lang.Interrupted
Exception
Public method found: public final void java.lang.Object.wait(long) throws java.lang.Interru
ptedException
Public method found: public final void java.lang.Object.wait(long,int) throws java.lang.Int
erruptedException
Public method found: public static void Program.main(java.lang.String[])
Public method found: public java.lang.String java.lang.Object.toString()
Public method found: public boolean java.lang.Object.equals(java.lang.Object)
Public method found: public int java.lang.Object.hashCode()

Method found: public static void Program.main(java.lang.String[])


Class modifiers: public synchronized

Super class name: java.lang.Object
*/
```

java.lang.Class.forName(String className)

Gets a bf6c9478-b86e-41e1-b8c9-48372912df0a object for a given class name.

J#

```
public static java.lang.Class forName(java.lang.String className)
```

Parameters

Parameter	Description
className	The fully qualified name (package plus class name) of a class. A ccdb1072-df6f-4c83-8eb8-ccd08a9e57d1 is thrown if this name is invalid.

Property Value/Return Value

A **bf6c9478-b86e-41e1-b8c9-48372912df0a** object for the given class name.

Example

```
// class_forname1.jsl

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get the Class object associated with
            // "java.lang.String".
            Class stringClass = Class.forName("java.lang.String");

            System.out.println("Class found: " +
                stringClass.getName());
        }
        catch (ClassNotFoundException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Class found: java.lang.String
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.forName(String assemblyName, String className, boolean absolutePath)

Gets a bf6c9478-b86e-41e1-b8c9-48372912df0a object for a given class name in an assembly.

J#

```
public static java.lang.Class forName(java.lang.String assemblyName, java.lang.String class
Name, boolean absolutePath)
```

Parameters

Parameter	Description
assemblyName	The assembly that contains the class.
className	The fully qualified name (package plus class name) of a class. A ccdb1072-df6f-4c83-8eb8-ccd08a9e57d1 is thrown if this name is invalid.
absolutePath	**true** if *assemblyName* is an absolute path; **false** otherwise.

Property Value/Return Value

A **bf6c9478-b86e-41e1-b8c9-48372912df0a** object for the given class name.

Example

```
// class_forname2.jsl

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get the Class object associated with
            // "java.lang.String".
            Class stringClass = Class.forName(
                "C:\\WINNT\\Microsoft.NET\\Framework\\v2.0.40903\\vjslib.dll",
                "java.lang.String", true);

            System.out.println("Class found: " +
                stringClass.getName());
        }
        catch (ClassNotFoundException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Class found: java.lang.String
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getClasses()

Gets all the public classes defined within the current class.

J#

```
public java.lang.Class[ ] getClasses()
```

Property Value/Return Value

An array of bf6c9478-b86e-41e1-b8c9-48372912df0a objects representing all the public classes defined within the current class.

Example

```
// class_getclasses.jsl

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get the Class object associated with "Program".
            Class thisClass = Class.forName("Program");

            // Get all the public Class objects associated with the
            // Program class.
            Class[] classes = thisClass.getClasses();
            for (int i = 0; i < classes.length; i++)
            {
                System.out.println("Class found: " +
                    classes[i].getName());
            }
        }
        catch (ClassNotFoundException ex)
        {
            System.out.println(ex.toString());
        }
    }

    public class InnerPublicClass
    {
        public InnerPublicClass()
        {
        }
    }

    private class InnerPrivateClass
    {
        public InnerPrivateClass()
        {
        }
    }
}

/*
Output:
Class found: Program$InnerClass
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getClassLoader()

Gets the bf6c9478-b86e-41e1-b8c9-48372912df0aLoader object associated with a bf6c9478-b86e-41e1-b8c9-48372912df0a object.

J#

```
public java.lang.ClassLoader getClassLoader()
```

Property Value/Return Value

The **bf6c9478-b86e-41e1-b8c9-48372912df0aLoader** object associated with a Class object, or **null** if the default system class was used.

Example

```
// class_getclassloader.jsl

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get the Class object associated with "java.lang.String".
            Class stringClass = Class.forName("java.lang.String");

            // Get the ClassLoader object associated with this Class.
            ClassLoader loader = stringClass.getClassLoader();

            if (loader == null)
            {
                System.out.println("The default system class " +
                    "was used.");
            }
            else
            {
                // Verify that this ClassLoader is associated with the
                // String class.
                Class loaderClass = loader.getClass();

                System.out.println("Class associated with ClassLoader: " +
                    loaderClass.getName());
            }
        }
        catch (ClassNotFoundException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The default system class was used.
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getComponentType()

Gets the component type of an array.

J#

```
public java.lang.Class getComponentType()
```

Property Value/Return Value

A bf6c9478-b86e-41e1-b8c9-48372912df0a object representing the component type of an array, or **null** if the type is not an array.

Example

```
// class_getcomponentyype.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Create a String object and an array of String objects.
        String str = "Hello, World!";
        String[] strArr = new String[] {
            "Hello", "World"
        };

        // Get the component type for the String.
        Class strClass = str.getClass();
        Class strClassComp = strClass.getComponentType();
        if (strClassComp != null)
        {
            System.out.println("String component type: " +
                strClassComp.getName());
        }
        else
        {
            System.out.println("String component type is null.");
        }

        // Get the component type for the array of String objects.
        Class strArrClass = strArr.getClass();
        Class strArrClassComp = strArrClass.getComponentType();
        if (strArrClassComp != null)
        {
            System.out.println("String array component type: " +
                strArrClassComp.getName());
        }
        else
        {
            System.out.println("String array component type " +
                "is null.");
        }
    }
}

/*
Output:
String component type is null.
String array component type: java.lang.String
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getConstructor(Class[] parameterTypes)

Gets the public constructor that takes the given arguments.

J#

```
public java.lang.reflect.Constructor getConstructor(java.lang.Class[ ] parameterTypes)
```

Parameters

Parameter	Description
parameterTypes	An array of bf6c9478-b86e-41e1-b8c9-48372912df0a objects representing the arguments to the constructor. A 5c106966-ba6e-4e68-8ed8-e4131d1bebff is thrown if there is no constructor with the given parameter types or if the constructor does not have public access.

Property Value/Return Value

A 6350a6f2-aaa4-4243-a87e-bb08980b51a8 object representing the constructor that accepts the given parameter types.

Example

```
// class_getconstructor.jsl

import java.lang.reflect.Constructor;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get the Class object associated with this class.
            Program program = new Program();
            Class progClass = program.getClass();

            // Find the constructor that only takes a String.
            Class[] ctorArgs1 = new Class[1];
            ctorArgs1[0] = String.class;
            Constructor strCtor = progClass.getConstructor(ctorArgs1);
            System.out.println("Constructor found: " +
                strCtor.toString());

            // Find the constructor that takes a String and an int.
            Class[] ctorArgs2 = new Class[2];
            ctorArgs2[0] = String.class;
            ctorArgs2[1] = Integer.class;
            Constructor strIntCtor = progClass.getConstructor(ctorArgs2);
            System.out.println("Constructor found: " +
                strIntCtor.toString());
        }
        catch (NoSuchMethodException ex)
        {
            System.out.println("Constructor doesn't exist or is " +
                "not public: " + ex.toString());
        }
    }

    public Program()
    {
    }

    public Program(String str)
    {
        this.str = str;
```

```
    }

    private Program(String str, Integer i)
    {
        this.str = str;
        this.i = i;
    }

    private String str = "Hello";
    private Integer i = new Integer(0);
}

/*
Output:
Constructor found: public Program(java.lang.String)
Constructor doesn't exist or is not public: java.lang.NoSuchMethodException
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getConstructors()

Gets all the public constructors for a given bf6c9478-b86e-41e1-b8c9-48372912df0a object.

J#

```
public java.lang.reflect.Constructor[ ] getConstructors()
```

Property Value/Return Value

An array of 6350a6f2-aaa4-4243-a87e-bb08980b51a8 objects representing all the public constructors for a given class.

Example

```
// class_getconstructors.jsl

import java.lang.reflect.Constructor;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get all the public constructors for ths class.
        Constructor[] ctors = progClass.getConstructors();
        for (int i = 0; i < ctors.length; i++)
        {
            System.out.println("Public constructor found: " +
                ctors[i].toString());
        }
    }

    public Program()
    {
    }

    public Program(String str)
    {
        this.str = str;
    }

    private Program(String str, Integer i)
    {
        this.str = str;
        this.i = i;
    }

    private String str = "Hello";
    private Integer i = new Integer(0);
}

/*
Output:
Public constructor found: public Program()
Public constructor found: public Program(java.lang.String)
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getDeclaredClasses()

Gets all the classes (public, private, and protected) defined within the current class.

J#

```
public java.lang.Class[ ] getDeclaredClasses()
```

Property Value/Return Value

An array of bf6c9478-b86e-41e1-b8c9-48372912df0a objects representing all the classes (public, private, and protected) defined within the current class.

Example

```
// class_getdeclaredclasses.jsl

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get the Class object associated with "Program".
            Class thisClass = Class.forName("Program");

            // Get all the Class objects associated with the
            // Program class.
            Class[] classes = thisClass.getDeclaredClasses();
            for (int i = 0; i < classes.length; i++)
            {
                System.out.println("Class found: " +
                    classes[i].getName());
            }
        }
        catch (ClassNotFoundException ex)
        {
            System.out.println(ex.toString());
        }
    }

    public class InnerPublicClass
    {
        public InnerPublicClass()
        {
        }
    }

    private class InnerPrivateClass
    {
        public InnerPrivateClass()
        {
        }
    }
}

/*
Output:
Class found: Program$InnerPublicClass
Class found: Program$InnerPrivateClass
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getDeclaredConstructor(Class[] parameterTypes)

Gets the constructor (public, private, or protected) that takes the given arguments.

J#

```
public java.lang.reflect.Constructor getDeclaredConstructor(java.lang.Class[ ] parameterTyp
es)
```

Parameters

Parameter	Description
parameterTypes	An array of bf6c9478-b86e-41e1-b8c9-48372912df0a objects representing the arguments to the constructor. A 5c106966-ba6e-4e68-8ed8-e4131d1bebff is thrown if there is no constructor with the given parameter types.

Property Value/Return Value

A 6350a6f2-aaa4-4243-a87e-bb08980b51a8 object representing the constructor that accepts the given parameter types.

Example

```
// class_getdeclaredconstructor.jsl

import java.lang.reflect.Constructor;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get the Class object associated with this class.
            Program program = new Program();
            Class progClass = program.getClass();

            // Find the constructor that only takes a String.
            Class[] ctorArgs1 = new Class[1];
            ctorArgs1[0] = String.class;
            Constructor strCtor =
                progClass.getDeclaredConstructor(ctorArgs1);
            System.out.println("Constructor found: " +
                strCtor.toString());

            // Find the constructor that takes a String and an int.
            Class[] ctorArgs2 = new Class[2];
            ctorArgs2[0] = String.class;
            ctorArgs2[1] = Integer.class;
            Constructor strIntCtor =
                progClass.getDeclaredConstructor(ctorArgs2);
            System.out.println("Constructor found: " +
                strIntCtor.toString());
        }
        catch (NoSuchMethodException ex)
        {
            System.out.println("Constructor doesn't exist: " +
                ex.toString());
        }
    }

    public Program()
    {
```

```
    }

    public Program(String str)
    {
        this.str = str;
    }

    private Program(String str, Integer i)
    {
        this.str = str;
        this.i = i;
    }

    private String str = "Hello";
    private Integer i = new Integer(0);
}

/*
Output:
Constructor found: public Program(java.lang.String)
Constructor found: private Program(java.lang.String,java.lang.Integer)
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getDeclaredConstructors()

Gets all the constructors (public, private, and protected) for a given bf6c9478-b86e-41e1-b8c9-48372912df0a object.

J#

```
public java.lang.reflect.Constructor[ ] getDeclaredConstructors()
```

Property Value/Return Value

An array of 6350a6f2-aaa4-4243-a87e-bb08980b51a8 objects representing all the constructors (public, private, and protected) for a given class.

Example

```
// class_getdeclaredconstructors.jsl

import java.lang.reflect.Constructor;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get all the constructors for ths class.
        Constructor[] ctors = progClass.getDeclaredConstructors();
        for (int i = 0; i < ctors.length; i++)
        {
            System.out.println("Constructor found: " +
                ctors[i].toString());
        }
    }

    public Program()
    {
    }

    public Program(String str)
    {
        this.str = str;
    }

    private Program(String str, Integer i)
    {
        this.str = str;
        this.i = i;
    }

    private String str = "Hello";
    private Integer i = new Integer(0);
}

/*
Output:
Constructor found: public Program()
Constructor found: public Program(java.lang.String)
Constructor found: private Program(java.lang.String,java.lang.Integer)
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getDeclaredField(String fieldName)

Gets the field (public, private, or protected) with the given name.

J#

```
public java.lang.reflect.Field getDeclaredField(java.lang.String fieldName)
```

Parameters

Parameter	Description
fieldName	The name of the field. A abfe0115-00b8-44a6-b2f8-927aa2a8cbcc is thrown if there is no field with the given name.

Property Value/Return Value

A 3d033269-e08f-4bd9-b84e-9d75a0cddd44 object representing the field of the given name.

Example

```
// class_getdeclaredfield.jsl

import java.lang.reflect.Field;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get the Class object associated with this class.
            Program program = new Program();
            Class progClass = program.getClass();

            // Get the field named str.
            Field strField = progClass.getDeclaredField("str");
            System.out.println("Field found: " + strField.toString());

            // Get the field named date.
            Field dateField = progClass.getDeclaredField("date");
            System.out.println("Field found: " + dateField.toString());

            // Get the field named i.
            Field iField = progClass.getDeclaredField("i");
            System.out.println("Field found: " + iField.toString());
        }
        catch (NoSuchFieldException ex)
        {
            System.out.println(ex.toString());
        }
    }

    public Program()
    {
    }

    public Program(String str, Date date, int i)
    {
        this.str = str;
        this.date = date;
        this.i = i;
    }
```

```
    public String str = "Hello";
    private Date date = new Date();
    protected int i = 0;
}

/*
Output:
Field found: public java.lang.String Program.str
Field found: private java.util.Date Program.date
Field found: protected int Program.i
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getDeclaredFields()

Gets all the fields (public, private, and protected) in the current class.

J#

```
public java.lang.reflect.Field[ ] getDeclaredFields()
```

Property Value/Return Value

An array of 3d033269-e08f-4bd9-b84e-9d75a0cddd44 objects representing all the fields (public, private, and protected) for a given class.

Example

```
// class_getdeclaredfields.jsl

import java.lang.reflect.Field;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get all the fields associated with this class.
        Field[] fields = progClass.getDeclaredFields();
        for (int i = 0; i < fields.length; i++)
        {
            System.out.println("Field found: " +
                fields[i].toString());
        }
    }

    public Program()
    {
    }

    public Program(String str, Date date, int i)
    {
        this.str = str;
        this.date = date;
        this.i = i;
    }

    public String str = "Hello";
    private Date date = new Date();
    protected int i = 0;
}

/*
Output:
Field found: public java.lang.String Program.str
Field found: private java.util.Date Program.date
Field found: protected int Program.i
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getDeclaredMethod(String methodName, Class[] parameterTypes)

Gets the method (public, private, or protected) with the given name and parameter types.

J#

```
public java.lang.reflect.Method getDeclaredMethod(java.lang.String methodName, java.lang.Cl
ass[ ] parameterTypes)
```

Parameters

Parameter	Description
methodName	The name of the method. A 5c106966-ba6e-4e68-8ed8-e4131d1bebff is thrown if there is no method with the given name and parameter types.
parameterTypes	An array of bf6c9478-b86e-41e1-b8c9-48372912df0a objects representing the arguments to the method. A **5c106966-ba6e-4e68-8ed8-e4131d1bebff** is thrown if there is no method with the given name and parameter types.

Property Value/Return Value

A c92d0bb6-52bf-4d52-8dc2-a8a093c211f0 object representing the method of the given name and parameter types.

Example

```
// class_getdeclaredmethod.jsl

import java.lang.reflect.Method;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get the Class object associated with this class.
            Program program = new Program();
            Class progClass = program.getClass();

            // Get the method named sayHello.
            Method helloMethod = progClass.getDeclaredMethod(
                "sayHello", null);
            System.out.println("Method found: " +
                helloMethod.toString());

            // Get the method named setStr.
            Class[] args1 = new Class[1];
            args1[0] = String.class;
            Method strMethod = progClass.getDeclaredMethod(
                "setStr", args1);
            System.out.println("Method found: " + strMethod.toString());

            // Get the method named setDate.
            Class[] args2 = new Class[1];
            args2[0] = Date.class;
            Method dateMethod = progClass.getDeclaredMethod(
                "setDate", args2);
            System.out.println("Method found: " + dateMethod.toString());

            // Get the method named setI.
```

```
            Class[] args3 = new Class[1];
            args3[0] = Integer.TYPE;
            Method iMethod = progClass.getDeclaredMethod(
                "setI", args3);
            System.out.println("Method found: " + iMethod.toString());
        }
        catch (NoSuchMethodException ex)
        {
            System.out.println(ex.toString());
        }
    }

    public Program()
    {
    }

    public String sayHello()
    {
        return "Hello!";
    }

    public void setStr(String str)
    {
        this.str = str;
    }

    private void setDate(Date date)
    {
        this.date = date;
    }

    private void setI(int i)
    {
        this.i = i;
    }

    public String str = "Hello";
    private Date date = new Date();
    protected int i = 0;
}

/*
Output:
Method found: public java.lang.String Program.sayHello()
Method found: public void Program.setStr(java.lang.String)
Method found: private void Program.setDate(java.util.Date)
Method found: private void Program.setI(int)
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getDeclaredMethods()

Gets all the methods (public, private, or protected) for the current class.

J#

```
public java.lang.reflect.Method[ ] getDeclaredMethods()
```

Property Value/Return Value

An array of c92d0bb6-52bf-4d52-8dc2-a8a093c211f0 objects representing all the methods (public, private, and protected) for a given class.

Example

```
// class_getdeclaredmethods.jsl

import java.lang.reflect.Method;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get all the methods associated with this class.
        Method[] methods = progClass.getDeclaredMethods();
        for (int i = 0; i < methods.length; i++)
        {
            System.out.println("Method found: " +
                methods[i].toString());
        }
    }

    public Program()
    {
    }

    public String sayHello()
    {
        return "Hello!";
    }

    public void setStr(String str)
    {
        this.str = str;
    }

    private void setDate(Date date)
    {
        this.date = date;
    }

    private void setI(int i)
    {
        this.i = i;
    }

    public String str = "Hello";
    private Date date = new Date();
    protected int i = 0;
}
```

```
/*
Output:
Method found: public static void Program.main(java.lang.String[])
Method found: public java.lang.String Program.sayHello()
Method found: public void Program.setStr(java.lang.String)
Method found: private void Program.setDate(java.util.Date)
Method found: private void Program.setI(int)
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getDeclaringClass()

Gets the class that declares an entity such as a field, method, or constructor.

J#

```
public java.lang.Class getDeclaringClass()
```

Property Value/Return Value

A bf6c9478-b86e-41e1-b8c9-48372912df0a object representing the class that declares an entity such as a field, method, or constructor.

Example

```
// class_getdeclaringclass.jsl

import java.lang.reflect.Method;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with the class String.
        Class strClass = String.class;

        // Get all methods for class String.
        Method[] methods = strClass.getMethods();
        for (int i = 0; i < methods.length; i++)
        {
            Class declaring = methods[i].getDeclaringClass();
            System.out.println("Method: " + methods[i].toString() +
                " Declaring class: " + declaring.toString());
        }
    }
}

/*
Output:
Method: public final java.lang.Class java.lang.Object.getClass() Declaring class: class jav
a.lang.Object
Method: public final void java.lang.Object.notify() Declaring class: class java.lang.Object
Method: public final void java.lang.Object.notifyAll() Declaring class: class java.lang.Obj
ect
Method: public final void java.lang.Object.wait() throws java.lang.InterruptedException Dec
laring class: class java.lang.Object
Method: public final void java.lang.Object.wait(long) throws java.lang.InterruptedException
Declaring class: class java.lang.Object
Method: public final void java.lang.Object.wait(long,int) throws java.lang.InterruptedExcep
tion Declaring class: class java.lang.Object
Method: public final java.lang.String java.lang.String.toString() Declaring class: class ja
va.lang.String
Method: public char java.lang.String.charAt(int) Declaring class: class java.lang.String
Method: public int java.lang.String.compareTo(java.lang.String) Declaring class: class java
.lang.String
Method: public java.lang.String java.lang.String.concat(java.lang.String) Declaring class:
class java.lang.String
Method: public static java.lang.String java.lang.String.copyValueOf(char[],int,int) Declari
ng class: class java.lang.String
Method: public static java.lang.String java.lang.String.copyValueOf(char[]) Declaring class
: class java.lang.String
Method: public boolean java.lang.String.endsWith(java.lang.String) Declaring class: class j
ava.lang.String
Method: public boolean java.lang.String.equals(java.lang.Object) Declaring class: class jav
a.lang.String
```

```
Method: public boolean java.lang.String.equals(java.lang.Object) Declaring class: class jav
a.lang.String
Method: public boolean java.lang.String.equalsIgnoreCase(java.lang.String) Declaring class:
class java.lang.String
Method: public byte[] java.lang.String.getBytes() Declaring class: class java.lang.String
Method: public byte[] java.lang.String.getBytes(java.lang.String) throws java.io.Unsupporte
dEncodingException Declaring class: class java.lang.String
Method: public void java.lang.String.getBytes(int,int,byte[],int) Declaring class: class ja
va.lang.String
Method: public void java.lang.String.getChars(int,int,char[],int) Declaring class: class ja
va.lang.String
Method: public int java.lang.String.hashCode() Declaring class: class java.lang.String
Method: public int java.lang.String.indexOf(int,int) Declaring class: class java.lang.Strin
g
Method: public int java.lang.String.indexOf(int) Declaring class: class java.lang.String
Method: public int java.lang.String.indexOf(java.lang.String,int) Declaring class: class ja
va.lang.String
Method: public int java.lang.String.indexOf(java.lang.String) Declaring class: class java.l
ang.String
Method: public java.lang.String java.lang.String.intern() Declaring class: class java.lang.
String
Method: public int java.lang.String.lastIndexOf(int,int) Declaring class: class java.lang.S
tring
Method: public int java.lang.String.lastIndexOf(int) Declaring class: class java.lang.Strin
g
Method: public int java.lang.String.lastIndexOf(java.lang.String,int) Declaring class: clas
s java.lang.String
Method: public int java.lang.String.lastIndexOf(java.lang.String) Declaring class: class ja
va.lang.String
Method: public int java.lang.String.length() Declaring class: class java.lang.String
Method: public boolean java.lang.String.regionMatches(int,java.lang.String,int,int) Declari
ng class: class java.lang.String
Method: public boolean java.lang.String.regionMatches(boolean,int,java.lang.String,int,int)
Declaring class: class java.lang.String
Method: public java.lang.String java.lang.String.replace(char,char) Declaring class: class
java.lang.String
Method: public boolean java.lang.String.startsWith(java.lang.String) Declaring class: class
java.lang.String
Method: public boolean java.lang.String.startsWith(java.lang.String,int) Declaring class: c
lass java.lang.String
Method: public java.lang.String java.lang.String.substring(int) Declaring class: class java
.lang.String
Method: public java.lang.String java.lang.String.substring(int,int) Declaring class: class
java.lang.String
Method: public char[] java.lang.String.toCharArray() Declaring class: class java.lang.Strin
g
Method: public java.lang.String java.lang.String.toString() Declaring class: class java.lan
g.String
Method: public java.lang.String java.lang.String.toLowerCase() Declaring class: class java.
lang.String
Method: public java.lang.String java.lang.String.toLowerCase(java.util.Locale) Declaring cl
ass: class java.lang.String
Method: public java.lang.String java.lang.String.toUpperCase() Declaring class: class java.
lang.String
Method: public java.lang.String java.lang.String.toUpperCase(java.util.Locale) Declaring cl
ass: class java.lang.String
Method: public java.lang.String java.lang.String.trim() Declaring class: class java.lang.St
ring
Method: public static java.lang.String java.lang.String.valueOf(boolean) Declaring class: c
lass java.lang.String
Method: public static java.lang.String java.lang.String.valueOf(char) Declaring class: clas
s java.lang.String
Method: public static java.lang.String java.lang.String.valueOf(double) Declaring class: cl
ass java.lang.String
Method: public static java.lang.String java.lang.String.valueOf(float) Declaring class: cla
ss java.lang.String
Method: public static java.lang.String java.lang.String.valueOf(int) Declaring class: class
java.lang.String
```

```
Method: public static java.lang.String java.lang.String.valueOf(long) Declaring class: clas
s java.lang.String
Method: public static java.lang.String java.lang.String.valueOf(java.lang.Object) Declaring
class: class java.lang.String
Method: public static java.lang.String java.lang.String.valueOf(char[]) Declaring class: cl
ass java.lang.String
Method: public static java.lang.String java.lang.String.valueOf(char[],int,int) Declaring c
lass: class java.lang.String
Method: public int java.lang.String.compareTo(java.lang.Object) Declaring class: class java
.lang.String
Method: public int java.lang.String.compareToIgnoreCase(java.lang.String) Declaring class:
class java.lang.String
Method: public int java.lang.Object.hashCode() Declaring class: class java.lang.Object
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getField(String fieldName)

Gets the public field with the given name.

J#

```
public java.lang.reflect.Field getField(java.lang.String fieldName)
```

Parameters

Parameter	Description
fieldName	The name of the field. A abfe0115-00b8-44a6-b2f8-927aa2a8cbcc is thrown if there is no field with the given name or if the field is not public.

Property Value/Return Value

A 3d033269-e08f-4bd9-b84e-9d75a0cddd44 object representing the public field of the given name.

Example

```
// class_getfield.jsl

import java.lang.reflect.Field;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get the field named str.
        try
        {
            Field strField = progClass.getField("str");
            System.out.println("Public field found: " +
                strField.toString());
        }
        catch (NoSuchFieldException ex)
        {
            System.out.println("Field either doesn't exist or is " +
                "not public: " + ex.toString());
        }

        // Get the field named date.
        try
        {
           Field dateField = progClass.getField("date");
           System.out.println("Public field found: " +
               dateField.toString());
        }
        catch (NoSuchFieldException ex)
        {
            System.out.println("Field either doesn't exist or is " +
                "not public: " + ex.toString());
        }

        // Get the field named i.
        try
        {
            Field iField = progClass.getField("i");
            System.out.println("Public field found: " +
```

```
                iField.toString());
        }
        catch (NoSuchFieldException ex)
        {
            System.out.println("Field either doesn't exist or is " +
                "not public: " + ex.toString());
        }
    }

    public Program()
    {
    }

    public Program(String str, Date date, int i)
    {
        this.str = str;
        this.date = date;
        this.i = i;
    }

    public String str = "Hello";
    private Date date = new Date();
    protected int i = 0;
}

/*
Output:
Public field found: public java.lang.String Program.str
Field either doesn't exist or is not public: java.lang.NoSuchFieldException: date
Field either doesn't exist or is not public: java.lang.NoSuchFieldException: i
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getFields()

Gets all the public fields in the current class.

J#

```
public java.lang.reflect.Field[ ] getFields()
```

Property Value/Return Value

An array of 3d033269-e08f-4bd9-b84e-9d75a0cddd44 objects representing all the public fields for a given class.

Example

```
// class_getfields.jsl

import java.lang.reflect.Field;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get all the public fields associated with this class.
        Field[] fields = progClass.getFields();
        for (int i = 0; i < fields.length; i++)
        {
            System.out.println("Public field found: " +
                fields[i].toString());
        }
    }

    public Program()
    {
    }

    public Program(String str, Date date, int i)
    {
        this.str = str;
        this.date = date;
        this.i = i;
    }

    public String str = "Hello";
    private Date date = new Date();
    protected int i = 0;
}

/*
Output:
Public field found: public java.lang.String Program.str
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getInterfaces()

Gets all the interfaces implemented by a class.

J#

```
public java.lang.Class[ ] getInterfaces()
```

Property Value/Return Value

An array of bf6c9478-b86e-41e1-b8c9-48372912df0a objects representing all the interfaces that a class implements.

Example

```
// class_getinterfaces.jsl

public class Program implements Comparable
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get the interfaces associated with this class.
        Class[] interfaces = progClass.getInterfaces();
        for (int i = 0; i < interfaces.length; i++)
        {
            System.out.println("Interface found: " +
                interfaces[i].toString());
        }
    }

    public int compareTo(Object o)
    {
        if (o instanceof Program)
        {
            Program p = (Program)o;

            return this.str.compareTo(p.str);
        }

        throw new IllegalArgumentException();
    }

    private String str = "Hello";
}

/*
Output:
Interface found: interface java.lang.Comparable
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getMethod(String methodName, Class[] parameterTypes)

Gets the public method with the given name and parameter types.

J#

```
public java.lang.reflect.Method getMethod(java.lang.String methodName, java.lang.Class[ ] p
arameterTypes)
```

Parameters

Parameter	Description
methodName	The name of the method. A 5c106966-ba6e-4e68-8ed8-e4131d1bebff is thrown if there is no method with the given name and parameter types or if the method is not public.
parameterTypes	An array of bf6c9478-b86e-41e1-b8c9-48372912df0a objects representing the arguments to the method. A **5c106966-ba6e-4e68-8ed8-e4131d1bebff** is thrown if there is no method with the given name and parameter types or if the method is not public.

Property Value/Return Value

A Method object representing the public method of the given name and parameter types.

Example

```
// class_getmethod.jsl

import java.lang.reflect.Method;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        try
        {
            // Get the method named sayHello.
            Method helloMethod = progClass.getMethod(
                "sayHello", null);
            System.out.println("Public method found: " +
                helloMethod.toString());
        }
        catch (NoSuchMethodException ex)
        {
            System.out.println("Method either doesn't exist " +
                "or is not public: " + ex.toString());
        }

        try
        {
            // Get the method named setStr.
            Class[] args1 = new Class[1];
            args1[0] = String.class;
            Method strMethod = progClass.getMethod(
                "setStr", args1);
            System.out.println("Public method found: " +
```

```
                strMethod.toString());
        }
        catch (NoSuchMethodException ex)
        {
            System.out.println("Method either doesn't exist " +
                "or is not public: " + ex.toString());
        }

        try
        {
            // Get the method named setDate.
            Class[] args2 = new Class[1];
            args2[0] = Date.class;
            Method dateMethod = progClass.getMethod(
                "setDate", args2);
            System.out.println("Public method found: " +
                dateMethod.toString());
        }
        catch (NoSuchMethodException ex)
        {
            System.out.println("Method either doesn't exist " +
                "or is not public: " + ex.toString());
        }

        try
        {
            // Get the method named setI.
            Class[] args3 = new Class[1];
            args3[0] = Integer.TYPE;
            Method iMethod = progClass.getMethod(
                "setI", args3);
            System.out.println("Public method found: " +
                iMethod.toString());
        }
        catch (NoSuchMethodException ex)
        {
            System.out.println("Method either doesn't exist " +
                "or is not public: " + ex.toString());
        }
    }

    public Program()
    {
    }

    public String sayHello()
    {
        return "Hello!";
    }

    public void setStr(String str)
    {
        this.str = str;
    }

    private void setDate(Date date)
    {
        this.date = date;
    }

    private void setI(int i)
    {
        this.i = i;
    }

    public String str = "Hello";
    private Date date = new Date();
    protected int i = 0;
```

```
}

/*
Output:
Public method found: public java.lang.String Program.sayHello()
Public method found: public void Program.setStr(java.lang.String)
Method either doesn't exist or is not public: java.lang.NoSuchMethodException: setDate
Method either doesn't exist or is not public: java.lang.NoSuchMethodException: setI
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getMethods()

Gets all the public methods for the current class.

J#

```
public java.lang.reflect.Method[ ] getMethods()
```

Parameters

Property Value/Return Value

An array of c92d0bb6-52bf-4d52-8dc2-a8a093c211f0 objects representing all the public methods for a given class.

Example

```
// class_getmethods.jsl

import java.lang.reflect.Method;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get the public methods associated with this class.
        Method[] methods = progClass.getMethods();
        for (int i = 0; i < methods.length; i++)
        {
            System.out.println("Public method found: " +
                methods[i].toString());
        }
    }

    public Program()
    {
    }

    public String sayHello()
    {
        return "Hello!";
    }

    public void setStr(String str)
    {
        this.str = str;
    }

    private void setDate(Date date)
    {
        this.date = date;
    }

    private void setI(int i)
    {
        this.i = i;
    }

    public String str = "Hello";
    private Date date = new Date();
    protected int i = 0;
}
```

```
/*
Output:
Public method found: public final java.lang.Class java.lang.Object.getClass()
Public method found: public final void java.lang.Object.notify()
Public method found: public final void java.lang.Object.notifyAll()
Public method found: public final void java.lang.Object.wait() throws java.lang.Interrupted
Exception
Public method found: public final void java.lang.Object.wait(long) throws java.lang.Interru
ptedException
Public method found: public final void java.lang.Object.wait(long,int) throws java.lang.Int
erruptedException
Public method found: public static void Program.main(java.lang.String[])
Public method found: public java.lang.String Program.sayHello()
Public method found: public void Program.setStr(java.lang.String)
Public method found: public java.lang.String java.lang.Object.toString()
Public method found: public boolean java.lang.Object.equals(java.lang.Object)
Public method found: public int java.lang.Object.hashCode()
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getModifiers()

Gets all the modifiers for a class declaration.

J#

```
public int getModifiers()
```

Property Value/Return Value

An integer representing the modifiers for the class declaration. Use the d408f6cd-0896-4706-a92b-5722881b88ea class to interpret this value.

Example

```
// class_getmodifiers.jsl

import java.lang.reflect.Modifier;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get the modifiers for this class.
        int i = progClass.getModifiers();
        String mods = Modifier.toString(i);
        System.out.println("Class modifiers: " + mods);

    }
}

/*
Output:
Class modifiers: public synchronized
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getName()

Gets the name of a class.

J#

```
public java.lang.String getName()
```

Property Value/Return Value

The name of the class.

Example

```
// class_getname.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get the name of this class.
        String name = progClass.getName();
        System.out.println("Class name: " + name);
    }
}

/*
Output:
Class name: Program
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.getResource(String resourceName)

This method is not supported by J#.

J#

```
public java.net.URL getResource(java.lang.String resourceName)
```

Parameters

Parameter	Description
resourceName	This parameter is not supported by J#.

See Also

Reference

java.lang.Class

java.lang.Class.getResourceAsStream(String resourceName)

This method is not supported by J#.

J#

```
public java.io.InputStream getResourceAsStream(java.lang.String resourceName)
```

See Also

Reference

java.lang.Class

java.lang.Class.getSigners()

Gets the array of signers that were set for this class, if any, by calling ClassLoader.setSigners(). A signer is an object that is used to sign the class for security.

J#

```
public java.lang.Object[ ] getSigners()
```

Property Value/Return Value

An array of signers that were set for this class, if any.

Remarks

NOTE: J# does not support custom class loaders or defineClass() methods. Hence, this method will always return a **null** array.

See Also

Reference

java.lang.Class

java.lang.Class.getSuperclass()

Gets the super class (base class) of the current class.

J#

```
public java.lang.Class getSuperclass()
```

Property Value/Return Value

A bf6c9478-b86e-41e1-b8c9-48372912df0a object representing the super (base) class of the current class.

Example

```
// class_getsuperclass.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get the super class of this class.
        Class superClass = progClass.getSuperclass();
        System.out.println("Super class name: " +
            superClass.getName());
    }
}

/*
Output:
Super class name: java.lang.Object
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.isArray()

Determines whether a class is an array.

J#

```
public boolean isArray()
```

Property Value/Return Value

true if the class is an array; **false** otherwise.

Example

```
// class_isarray.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Create a String object and an array of String objects.
        String str = "Hello, World!";
        String[] strArr = new String[] {
            "Hello", "World"
        };

        // Is the String class an array?
        Class strClass = str.getClass();
        boolean strIsArray = strClass.isArray();
        if (strIsArray)
        {
            System.out.println("Class " + strClass.getName() +
                " is an array.");
        }
        else
        {
            System.out.println("Class " + strClass.getName() +
                " is NOT an array.");
        }

        // Is the String array class an array?
        Class strArrClass = strArr.getClass();
        boolean strArrIsArray = strArrClass.isArray();
        if (strArrIsArray)
        {
            System.out.println("Class " + strArrClass.getName() +
                " is an array.");
        }
        else
        {
            System.out.println("Class " + strArrClass.getName() +
                " is NOT an array.");
        }
    }
}

/*
Output:
Class java.lang.String is NOT an array.
Class [Ljava.lang.String; is an array.
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.isAssignableFrom(Class cls)

Determines whether one class can be assigned from another class.

J#

```
public boolean isAssignableFrom(java.lang.Class cls)
```

Parameters

Parameter	Description
cls	A bf6c9478-b86e-41e1-b8c9-48372912df0a object representing the class to verify if it can be assigned to the current class.

Property Value/Return Value

true if *cls* can be assigned to the current class; **false** otherwise.

Example

```
// class_isassignablefrom.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get the Class object associated with BaseClass.
        Class baseClassClass = BaseClass.class;

        // Is Program assignable from BaseClass?
        boolean check1 = progClass.isAssignableFrom(baseClassClass);
        System.out.println("Program is assignable from BaseClass? " +
            check1);

        // Is BaseClass assignable from Program?
        boolean check2 = baseClassClass.isAssignableFrom(progClass);
        System.out.println("BaseClass is assignable from Program? " +
            check2);
    }
}

public class BaseClass extends Program
{
    public BaseClass()
    {
    }
}

/*
Output:
Program is assignable from BaseClass? true
BaseClass is assignable from Program? false
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.isInstance(Object obj)

Determines whether an object is an instance of the current class.

J#

```
public boolean isInstance(java.lang.Object obj)
```

Parameters

Parameter	Description
obj	The object to verify if it is an instance of the current class.

Property Value/Return Value

true if *obj* is an instance of the current class; **false** otherwise.

Remarks

This method is equivalent to calling `obj instanceof ClassName`.

Example

```
// class_isinstance.jsl

import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with Integer.
        Class intClass = Integer.class;

        // Create various objects.
        String str = "Hello";
        Date date = new Date();
        Integer i = new Integer(10);

        // Is str an instance of class Integer?
        boolean check1 = intClass.isInstance(str);
        System.out.println("str is an Integer? " + check1);

        // Is date an instance of class Integer?
        boolean check2 = intClass.isInstance(date);
        System.out.println("date is an Integer? " + check2);

        // Is i an instance of class Integer?
        boolean check3 = intClass.isInstance(i);
        System.out.println("i is an Integer? " + check3);
    }
}

/*
Output:
str is an Integer? false
date is an Integer? false
i is an Integer? true
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.isInterface()

Determines whether the current bf6c9478-b86e-41e1-b8c9-48372912df0a object represents an interface.

J#

```
public boolean isInterface()
```

Property Value/Return Value

true if the current **bf6c9478-b86e-41e1-b8c9-48372912df0a** object represents an interface; **false** otherwise.

Example

```
// class_isinterface.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get the Class object associated with ISayHello.
        Class iSayHelloClass = ISayHello.class;

        // Is Program an interface?
        boolean check1 = progClass.isInterface();
        System.out.println("Program is an interface? " +
            check1);

        // Is ISayHello an interface?
        boolean check2 = iSayHelloClass.isInterface();
        System.out.println("ISayHello is an interface? " +
            check2);
    }
}

public interface ISayHello
{
    public String sayHello();
}

/*
Output:
Program is an interface? false
ISayHello is an interface? true
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.isPrimitive()

Determines whether the current bf6c9478-b86e-41e1-b8c9-48372912df0a object represents a primitive type.

J#

```
public boolean isPrimitive()
```

Property Value/Return Value

true if the current **bf6c9478-b86e-41e1-b8c9-48372912df0a** object represents a primitive type (such as **int**, **long**, **void**); **false** otherwise.

Example

```
// class_isprimitive.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Get the Class object associated with this class.
        Program program = new Program();
        Class progClass = program.getClass();

        // Get the Class object associated with an integer.
        int i = 0;
        Class iClass = int.class;

        // Is Program a primitive type?
        boolean check1 = progClass.isPrimitive();
        System.out.println("program is a primitive type? " +
            check1);

        // Is i an primitive type?
        boolean check2 = iClass.isPrimitive();
        System.out.println("i is a primitive type? " +
            check2);
    }
}

/*
Output:
program is a primitive type? false
i is a primitive type? true
*/
```

See Also

Reference

java.lang.Class

java.lang.Class.newInstance()

Creates a new instance of the object represented by the current bf6c9478-b86e-41e1-b8c9-48372912df0a object.

J#

```
public java.lang.Object newInstance()
```

Property Value/Return Value

A new instance of the object represented by the current **bf6c9478-b86e-41e1-b8c9-48372912df0a** object. An ab3181f6-40a0-4dcd-8ddc-575e4db64b05 will be thrown if the object cannot be instantiated. An 4dd49557-ba4e-475c-872d-6c06670d34b7 will be thrown if the caller does not have access to the object.

Example

```
// class_newinstance.jsl

import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Create a Date object for time now.
            Date now = new Date();
            Class dateClass = now.getClass();
            System.out.println("The time now is: " + now.toString());

            // Create another Date object using the Class for Date.
            Object o = dateClass.newInstance();
            if (o instanceof Date)
            {
                Date aLittleLater = (Date)o;
                System.out.println("The time a little later is: " +
                    aLittleLater.toString());
            }
        }
        catch (InstantiationException ex)
        {
            System.out.println(ex.toString());
        }
        catch (IllegalAccessException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Sample Output:
The time now is: Thu Oct 28 13:56:34 PDT 2004
The time a little later is: Thu Oct 28 13:56:35 PDT 2004
*/
```

See Also

Reference

java.lang.Class

java.lang.Object

Represents the base class for all J# objects. All classes in J# inherently derive from **b33599ff-0daf-4116-b5d8-0649793ac6b0** and inherit all its functionality.

J#

```
public class Object extends System.Object
```

Example

```
// object_overview.jsl

public class MyObject extends Object implements Runnable
{
    public static void main(String[] args)
    {
        try
        {
            // The object being used to call wait and notify on
            // must be synchronized.
            synchronized (obj)
            {
                // Create a secondary thread used to call notify.
                Thread thr = new Thread(obj);
                thr.start();

                // Wait for the secondary thread to call notify.
                System.out.println("waiting...");
                obj.wait();
                System.out.println("done waiting!");

                // Join the secondary thread.
                thr.join();
            }
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }

        // Get the Class associated with MyObject.
        Class myClass = obj.getClass();

        // Print out the type of obj.
        System.out.println("obj is of type " +
            myClass.getName());

        // Get the hash code associated with MyObject.
        int hash = obj.hashCode();

        // Print out the hash code for obj.
        System.out.println("The hash code for obj is " +
            hash);

        // Print out the string representation for obj.
        System.out.println("The string representation for obj is " +
            obj.toString());

        // Determine if two MyObject objects are equal.
        boolean equal = obj.equals(obj2);
        System.out.println("obj equals obj2? " + equal);
    }

    // Implements Runnable.run()
```

```
    public void run()
    {
        // The object being used to call wait and notify on
        // must be synchronized.
        synchronized (obj)
        {
            // Call notify on the primary thread so it can
            // stop waiting.
            System.out.println("\tnotifying...");
            obj.notify();
            System.out.println("\tdone notifying!");
        }
    }

    private static MyObject obj = new MyObject();
    private static MyObject obj2 = new MyObject();
}

/*
Sample Output:
waiting...
        notifying...
        done notifying!
done waiting!
obj is of type MyObject
The hash code for obj is 18643596
The string representation for obj is MyObject@11c7a8c
obj equals obj2? false
*/
```

See Also

Concepts

Visual J# Class Library

java.lang.Object.clone()

Creates a shallow copy of an b33599ff-0daf-4116-b5d8-0649793ac6b0.

J#

```
protected Object clone() throws CloneNotSupportedException
```

Property Value/Return Value

A new instance of an **b33599ff-0daf-4116-b5d8-0649793ac6b0** that is a shallow copy of the existing **b33599ff-0daf-4116-b5d8-0649793ac6b0**.

Example

```
// object_clone.jsl

public class MyObject extends Object
{
    public static void main(String[] args)
    {
        // Create three instance of MyObject objects.
        MyObject obj1 = new MyObject("Hello");
        MyObject obj2 = new MyObject("World");
        MyObject obj3 = null;
        try
        {
            obj3 = (MyObject)obj2.clone();
        }
        catch (CloneNotSupportedException ex)
        {
            System.out.println(ex.toString());
        }

        // Is obj1 equal to obj2?
        boolean equality = obj1.equals(obj2);
        System.out.println("obj1 equals obj2? " +
            equality);

        // Is obj2 equal to obj3?
        equality = obj2.equals(obj3);
        System.out.println("obj2 equals obj3? " +
            equality);
    }

    public MyObject(String s)
    {
        aString = s;
    }

    protected Object clone() throws CloneNotSupportedException
    {
        MyObject newObj = new MyObject(aString);
        return newObj;
    }

    // Two MyObject objects are equal if the aString
    // fields are equal.
    public boolean equals(Object o)
    {
        if (o instanceof MyObject)
        {
            MyObject myO = (MyObject)o;
            if (aString == myO.aString)
            {
```

```
                return true;
            }
        }

        return false;
    }

    private String aString = "";
}

/*
Output:
obj1 equals obj2? false
obj2 equals obj3? true
*/
```

See Also

Reference

java.lang.Object

java.lang.Object.equals(Object obj)

Determines whether two objects are equal.

J#

```
public boolean equals(Object obj)
```

Parameters

Parameter	Description
obj	An instance of an b33599ff-0daf-4116-b5d8-0649793ac6b0 to compare against the current **b33599ff-0daf-4116-b5d8-0649793ac6b0** for equality.

Property Value/Return Value

true if the two objects are equal; **false** otherwise.

Example

```
// object_equals.jsl

public class MyObject extends Object
{
    public static void main(String[] args)
    {
        // Create three instance of MyObject objects.
        MyObject obj1 = new MyObject("Hello");
        MyObject obj2 = new MyObject("World");
        MyObject obj3 = new MyObject("World");

        // Is obj1 equal to obj2?
        boolean equality = obj1.equals(obj2);
        System.out.println("obj1 equals obj2? " +
            equality);

        // Is obj2 equal to obj3?
        equality = obj2.equals(obj3);
        System.out.println("obj2 equals obj3? " +
            equality);
    }

    public MyObject(String s)
    {
        aString = s;
    }

    // Two MyObject objects are equal if the aString
    // fields are equal.
    public boolean equals(Object o)
    {
        if (o instanceof MyObject)
        {
            MyObject myO = (MyObject)o;
            if (aString == myO.aString)
            {
                return true;
            }
        }

        return false;
    }

    private String aString = "";
```

```
}

/*
Output:
obj1 equals obj2? false
obj2 equals obj3? true
*/
```

See Also

Reference

java.lang.Object

java.lang.Object.finalize()

Represents the method that is called once the garbage collector has determined that the b33599ff-0daf-4116-b5d8-0649793ac6b0 is no longer being accessed by any thread.

J#

```
protected void finalize() throws Throwable
```

Example

```
// There is no code example for this method.
```

See Also

Reference

java.lang.Object

java.lang.Object.getClass()

Gets the Class object representing the type of the current b33599ff-0daf-4116-b5d8-0649793ac6b0.

J#

```
public final Class getClass()
```

Property Value/Return Value

A bf6c9478-b86e-41e1-b8c9-48372912df0a object that represents the type of the current **b33599ff-0daf-4116-b5d8-0649793ac6b0**.

Example

```
// object_getclass.jsl

public class MyObject extends Object
{
    public static void main(String[] args)
    {
        // Create a new instance of MyObject.
        MyObject obj = new MyObject();

        // Get the Class associated with MyObject.
        Class myClass = obj.getClass();

        // Print out the type of obj.
        System.out.println("obj is of type " +
            myClass.getName());
    }
}

/*
Output:
obj is of type MyObject
*/
```

See Also

Reference

java.lang.Object

java.lang.Object.hashCode()

Generates a hash value that can be used to uniquely identify a particular b33599ff-0daf-4116-b5d8-0649793ac6b0.

J#

```
public int hashCode()
```

Property Value/Return Value

A hash value that uniquely identifies an **b33599ff-0daf-4116-b5d8-0649793ac6b0**.

Remarks

Subsequent calls to this method must consistently produce the same hash value.

Example

```
// object_hashcode.jsl

public class MyObject extends Object
{
    public static void main(String[] args)
    {
        // Create a new instance of MyObject.
        MyObject obj = new MyObject();

        // Get the hash code associated with MyObject.
        int hash = obj.hashCode();

        // Print out the hash code for obj.
        System.out.println("The hash code for obj is " +
            hash);
    }
}

/*
Sample Output:
The hash code for obj is 54267293
*/
```

See Also

Reference

java.lang.Object

java.lang.Object.notify()

Causes any thread that is currently waiting to be awakened.

J#

```
public final void notify()
```

Remarks

If more than one thread is waiting, the choice for which thread is to be woken is random.

Example

```
// object_notify.jsl

public class MyObject extends Object implements Runnable
{
    public static void main(String[] args)
    {
        try
        {
            // The object being used to call wait and notify on
            // must be synchronized.
            synchronized (obj)
            {
                // Create a secondary thread used to call notify.
                Thread thr = new Thread(obj);
                thr.start();

                // Wait for the secondary thread to call notify.
                System.out.println("waiting...");
                obj.wait();
                System.out.println("done waiting!");

                // Join the secondary thread.
                thr.join();
            }
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        // The object being used to call wait and notify on
        // must be synchronized.
        synchronized (obj)
        {
            // Call notify on the primary thread so it can
            // stop waiting.
            System.out.println("\tnotifying...");
            obj.notify();
            System.out.println("\tdone notifying!");
        }
    }

    private static MyObject obj = new MyObject();
}

/*
Output:
```

```
waiting...
        notifying...
        done notifying!
done waiting!
*/
```

See Also

Reference

java.lang.Object

java.lang.Object.notifyAll()

Causes all threads that are currently waiting to be awakened.

J#

```
public final void notifyAll()
```

Example

```
// object_notifyall.jsl

public class MyObject extends Object implements Runnable
{
    public static void main(String[] args)
    {
        try
        {
            // The object being used to call wait and notifyAll on
            // must be synchronized.
            synchronized (obj)
            {
                // Create a secondary thread used to call notifyAll.
                Thread thr = new Thread(obj);
                thr.start();

                // Wait for the secondary thread to call notifyAll.
                System.out.println("waiting...");
                obj.wait();
                System.out.println("done waiting!");

                // Join the secondary thread.
                thr.join();
            }
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        // The object being used to call wait and notifyAll on
        // must be synchronized.
        synchronized (obj)
        {
            // Call notifyAll on the primary thread so it can
            // stop waiting.
            System.out.println("\tnotifying...");
            obj.notifyAll();
            System.out.println("\tdone notifying!");
        }
    }

    private static MyObject obj = new MyObject();
}

/*
Output:
waiting...
        notifying...
        done notifying!
done waiting!
```

```
*/
```

See Also

Reference

java.lang.Object

java.lang.Object.toString()

Displays a human-readable representation of an b33599ff-0daf-4116-b5d8-0649793ac6b0.

J#

```
public String toString()
```

Property Value/Return Value

A human-readable representation of an **b33599ff-0daf-4116-b5d8-0649793ac6b0**.

Example

```
// object_tostring.jsl

public class MyObject extends Object
{
    public static void main(String[] args)
    {
        // Create a new instance of MyObject.
        MyObject obj = new MyObject();

        // Print out the string representation for obj.
        System.out.println("The string representation for obj is " +
            obj.toString());
    }
}

/*
Sample Output:
The string representation for obj is MyObject@33c0d9d
*/
```

See Also

Reference

java.lang.Object

java.lang.Object.wait()

Causes the currently executing thread to wait until another thread calls the notify method.

J#

```
public final void wait() throws InterruptedException
```

Example

```
// object_wait.jsl

public class MyObject extends Object implements Runnable
{
    public static void main(String[] args)
    {
        try
        {
            // The object being used to call wait and notify on
            // must be synchronized.
            synchronized (obj)
            {
                // Create a secondary thread used to call notify.
                Thread thr = new Thread(obj);
                thr.start();

                // Wait for the secondary thread to call notify.
                System.out.println("waiting...");
                obj.wait();
                System.out.println("done waiting!");

                // Join the secondary thread.
                thr.join();
            }
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        // The object being used to call wait and notify on
        // must be synchronized.
        synchronized (obj)
        {
            // Call notify on the primary thread so it can
            // stop waiting.
            System.out.println("\tnotifying...");
            obj.notify();
            System.out.println("\tdone notifying!");
        }
    }

    private static MyObject obj = new MyObject();
}

/*
Output:
waiting...
        notifying...
        done notifying!
done waiting!
```

```
*/
```

See Also

Reference

java.lang.Object

java.lang.Object.wait(long millis)

Causes the currently executing thread to wait until another thread calls the notify method or for the specified duration.

J#

```
public final void wait(long millis) throws InterruptedException
```

Parameters

Parameter	Description
millis	The number of milliseconds to wait.

Example

```
// object_wait_2.jsl

public class MyObject extends Object implements Runnable
{
    public static void main(String[] args)
    {
        try
        {
            // The object being used to call wait and notify on
            // must be synchronized.
            synchronized (obj)
            {
                // Create a secondary thread used to call notify.
                Thread thr = new Thread(obj);
                thr.start();

                // Wait either for the secondary thread to call
                // notify or for 5000 milliseconds.
                System.out.println("waiting...");
                obj.wait(5000);
                System.out.println("done waiting!");

                // Join the secondary thread.
                thr.join();
            }
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        // The object being used to call wait and notify on
        // must be synchronized.
        synchronized (obj)
        {
            // Call notify on the primary thread so it can
            // stop waiting.
            System.out.println("\tnotifying...");
            obj.notify();
            System.out.println("\tdone notifying!");
        }
    }

    private static MyObject obj = new MyObject();
}
```

```
/*
Output:
waiting...
        notifying...
        done notifying!
done waiting!
*/
```

See Also

Reference

java.lang.Object

java.lang.Object.wait(long millis, int nanos)

Causes the currently executing thread to wait until another thread calls the notify method or for the specified duration.

J#

```
public final void wait(long millis, int nanos) throws InterruptedException
```

Parameters

Parameter	Description
millis	The number of milliseconds to wait.
nanos	This parameter is ignored in J#.

Example

```
// object_wait_3.jsl

public class MyObject extends Object implements Runnable
{
    public static void main(String[] args)
    {
        try
        {
            // The object being used to call wait and notify on
            // must be synchronized.
            synchronized (obj)
            {
                // Create a secondary thread used to call notify.
                Thread thr = new Thread(obj);
                thr.start();

                // Wait either for the secondary thread to call
                // notify or for 10000 milliseconds.
                System.out.println("waiting...");
                obj.wait(10000, 0);
                System.out.println("done waiting!");

                // Join the secondary thread.
                thr.join();
            }
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        // The object being used to call wait and notify on
        // must be synchronized.
        synchronized (obj)
        {
            // Call notify on the primary thread so it can
            // stop waiting.
            System.out.println("\tnotifying...");
            obj.notify();
            System.out.println("\tdone notifying!");
        }
    }
```

```
    private static MyObject obj = new MyObject();
}

/*
Output:
waiting...
        notifying...
        done notifying!
done waiting!
*/
```

See Also

Reference

java.lang.Object

java.lang.reflect

Contains the classes and interfaces used to retrieve information on classes and objects.

Classes

Class	Description
Field	Provides access to the metadata and attributes of a field declared on a class. In addition, this class provides ways to get and set the value of a field dynamically.
Method	Contains methods to obtain reflection information on method names, parameters, modifiers, and return values. It also provides a way to dynamically invoke methods.

Interfaces

Interface	Description
Member	An interface that contains methods to retrieve information about members, such as fields or methods, and constructors.

Array Class

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.reflect.Array
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.lang.reflect.Array

See Also

Concepts

Array Members

java.lang.reflect Package

Array Members

The following tables list the members exposed by the Array type.

Public Constructors

Name	Description
Array	

Public Methods

Name	Description
equals	(inherited from Object)
get	
getBoolean	
getByte	
getChar	
getDouble	
getFloat	
hashCode	(inherited from Object)
getInt	
getLength	
getLong	
getShort	
getClass	(inherited from Object)
newInstance	Overloaded.
set	
setBoolean	
setByte	
setChar	
setDouble	
setFloat	
setInt	

setLong	
setShort	
toString	Overridden.

Protected Methods

MemberwiseClone

Performs a shallow copy of all members of the array.

See Also

Reference

Array Class

Concepts

java.lang.reflect Package

Array Constructor

Constructs an Arrays object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Arrays();
```

Example

```
// Construct an arrays object:
Arrays myArray = new Arrays();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Array Methods

Public Methods

Name	Description
equals	(inherited from Object)
get	
getBoolean	
getByte	
getChar	
getDouble	
getFloat	
hashCode	(inherited from Object)
getInt	
getLength	
getLong	
getShort	
getClass	(inherited from Object)
clone	
newInstance	Overloaded.
set	
setBoolean	
setByte	
setChar	
setDouble	
setFloat	
setInt	
setLong	
setShort	

toString	Overridden.

See Also

Reference

Array Class

Concepts

java.lang.reflect Package

Array.get Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Object get(
    java.lang.Object ar,
    int ix) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExcep
tion;
```

Parameters

ar

ix

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.getBoolean Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean getBoolean(
    java.lang.Object ar,
    int ix) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExcep
tion;
```

Parameters

ar

ix

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.getByte Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static byte getByte(
    java.lang.Object ar,
    int ix) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExcep
tion;
```

Parameters

ar

ix

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.getChar Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static char getChar(
    java.lang.Object ar,
    int ix) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExcep
tion;
```

Parameters

ar

ix

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.getDouble Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static double getDouble(
    java.lang.Object ar,
    int ix) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExcep
tion;
```

Parameters

ar

ix

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.getFloat Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static float getFloat(
    java.lang.Object ar,
    int ix) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExcep
tion;
```

Parameters

ar

ix

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.getInt Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static int getInt(
    java.lang.Object ar,
    int ix) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExcep
tion;
```

Parameters

ar

ix

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.getLength Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static int getLength(
    java.lang.Object ar) throws java.lang.IllegalArgumentException;
```

Parameters

ar

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.getLong Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static long getLong(
    java.lang.Object ar,
    int ix) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExcep
tion;
```

Parameters

ar

ix

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.getShort Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static short getShort(
    java.lang.Object ar,
    int ix) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExcep
tion;
```

Parameters

ar

ix

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.newInstance Method

Overload List

Name	Description
Array.newInstance (Class, int)	
Array.newInstance (Class, int[])	

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.newInstance Method (Class, Int32)

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Object newInstance(
    java.lang.Class componentType,
    int len) throws java.lang.NegativeArraySizeException;
```

Parameters

componentType

len

See Also
Reference
Array Class
Concepts
Array Members
java.lang.reflect Package

Array.newInstance Method (Class, Int32[ ])

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Object newInstance(
    java.lang.Class componentType,
    int len) throws java.lang.NegativeArraySizeException;
```

Parameters

componentType

len

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.set Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static void set(
    java.lang.Object ar,
    int ix,
    java.lang.Object val) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOu
tOfBoundsException;
```

Parameters

ar

ix

val

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.setBoolean Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static void setBoolean(
    java.lang.Object ar,
    int ix,
    boolean bval) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBound
sException;
```

Parameters

ar

ix

bval

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.setByte Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static void setByte(
    java.lang.Object ar,
    int ix,
    byte bval) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsEx
ception;
```

Parameters

ar

ix

bval

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.setChar Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static void setChar(
    java.lang.Object ar,
    int ix,
    char ch) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExce
ption;
```

Parameters

ar

ix

ch

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.setDouble Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static void setDouble(
    java.lang.Object ar,
    int ix,
    double dval) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBounds
Exception;
```

Parameters

ar

ix

dval

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.setFloat Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static void setFloat(
    java.lang.Object ar,
    int ix,
    float fval) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsE
xception;
```

Parameters

ar

ix

fval

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.setInt Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static void setInt(
    java.lang.Object ar,
    int ix,
    int ival) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsExc
eption;
```

Parameters

ar

ix

ival

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.setLong Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static void setLong(
    java.lang.Object ar,
    int ix,
    long lval) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsEx
ception;
```

Parameters

ar

ix

lval

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Array.setShort Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static void setShort(
    java.lang.Object ar,
    int ix,
    short sval) throws java.lang.IllegalArgumentException, java.lang.ArrayIndexOutOfBoundsE
xception;
```

Parameters

ar

ix

sval

See Also

Reference

Array Class

Concepts

Array Members

java.lang.reflect Package

Constructor Class

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.reflect.Constructor
    extends java.lang.Object
    implements java.lang.reflect.Member
```

Inheritance Hierarchy

java.lang.Object

java.lang.reflect.Constructor

See Also

Concepts

Constructor Members

java.lang.reflect Package

Constructor Members

The following tables list the members exposed by the Constructor type.

Public Methods

Name	Description
equals	Overridden.
getDeclaringClass	
getExceptionTypes	
hashCode	Overridden.
getModifiers	
getName	
getParameterTypes	
getClass	(inherited from Object)
newInstance	
toString	Overridden.

Protected Members

Name	Description
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

Constructor Class

Concepts

java.lang.reflect Package

Constructor Methods

Public Methods

Name	Description
equals	Overridden.
getDeclaringClass	
getExceptionTypes	
hashCode	Overridden.
getModifiers	
getName	
getParameterTypes	
getClass	(inherited from Object)
newInstance	
toString	Overridden.

Protected Methods

Name	Description
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

Constructor Class

Concepts

java.lang.reflect Package

Constructor.equals Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

Constructor Class

Concepts

Constructor Members

java.lang.reflect Package

Constructor.getDeclaringClass Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class getDeclaringClass();
```

See Also

Reference

Constructor Class

Concepts

Constructor Members

java.lang.reflect Package

Constructor.getExceptionTypes Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class[] getExceptionTypes();
```

See Also

Reference

Constructor Class

Concepts

Constructor Members

java.lang.reflect Package

Constructor.hashCode Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

Constructor Class

Concepts

Constructor Members

java.lang.reflect Package

Constructor.getModifiers Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public int getModifiers();
```

See Also

Reference

Constructor Class

Concepts

Constructor Members

java.lang.reflect Package

Constructor.getName Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getName();
```

See Also

Reference

Constructor Class

Concepts

Constructor Members

java.lang.reflect Package

Constructor.getParameterTypes Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class[] getParameterTypes();
```

See Also

Reference

Constructor Class

Concepts

Constructor Members

java.lang.reflect Package

Constructor.newInstance Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object newInstance(
    java.lang.Object[] initargs) throws java.lang.InstantiationException, java.lang.Illegal
AccessException, java.lang.IllegalArgumentException, java.lang.reflect.InvocationTargetExce
ption;
```

Parameters

initargs

See Also

Reference

Constructor Class

Concepts

Constructor Members

java.lang.reflect Package

Constructor.toString Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

Constructor Class

Concepts

Constructor Members

java.lang.reflect Package

Field Class

Provides access to the metadata and attributes of a field declared on a class. In addition, this class provides ways to get and set the value of a field dynamically.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.reflect.Field
    extends java.lang.Object
    implements java.lang.reflect.Member
```

Example

This example covers many of the Field class methods and uses some of the methods explained in the java.lang.Class class.

```
// Fld-General-example.jsl
// Field gGeneral example

import java.lang.reflect.*;

public class MyClass
{
    private int myIntField;
    private float myFloatField;
    private Object myObjectField;

    public static void main(String[] args) throws SecurityException
    {
        // Declare and initialize a Class object:
        Class mc = MyClass.class;

        // Declare the array of fields:
        Field[] fields = mc.getDeclaredFields();

        // Get the class name
        String name = mc.getName();

        // Display the class name:
        System.out.println("The class name: " + name);

        // Dipslay all the declared fields:
        System.out.println("The fields are:");
        for (int i = 0; i<fields.length; i++)
           System.out.println(fields[i]);
    }
}

/*
Output:
The class name: MyClass
The fields are:
private int MyClass.myIntField
private float MyClass.myFloatField
private java.lang.Object MyClass.myObjectField
*/
```

Remarks

Various methods of the java.lang.Class class, such as java.lang.Class.getField(String fieldName), java.lang.Class.getFields(), java.lang.Class.getDeclaredField(String fieldName), and java.lang.Class.getDeclaredFields() return instances of Field objects corresponding to the fields of a class.

Inheritance Hierarchy

java.lang.Object

  java.lang.reflect.Field

See Also

Concepts

Field Members

java.lang.reflect Package

Field Members

Provides access to the metadata and attributes of a field declared on a class. In addition, this class provides ways to get and set the value of a field dynamically.

The following tables list the members exposed by the Field type.

Public Methods

Name	Description
equals	Overridden. Compares the current Field to a specified object.
get	Returns the value of a specific Field in the current object.
getBoolean	Retrieves the value of a static or instance field of the type boolean.
getByte	Retrieves the value of a static or instance field of the type byte.
getChar	Retrieves the value of a static or instance field of the type char or of another type that can be converted to char.
getDeclaringClass	Returns the name of the class that declares the current field object.
getDouble	Returns the value of a double field object or of another primitive type that can be converted to double.
getFloat	Returns the value of a float field object or of another primitive type that can be converted to float.
hashCode	Overridden. Returns the hash code of a Field object.
getInt	Returns the value of a int field object or of another primitive type that can be converted to int.
getLong	Returns the value of a long field object or of another primitive type that can be converted to long.
getModifiers	Returns an int that represents the modifiers for the field in the current Field object.
getName	Returns the name of the field of the current Field object.
getShort	Returns the value of a short field object or of another primitive type that can be converted to short.
getType	Returns the field type of the current field object.
getClass	(inherited from Object)
set	Sets the value of the current field of the specified object to a specified value.
setBoolean	Sets the value of a boolean field on the current object.
setByte	Sets the value of a byte field on the current object.
setChar	Sets the value of a char field on the current object.
setDouble	Sets the value of a double field on the current object.
setFloat	Sets the value of a float field on the current object.

setInt	Sets the value of an int field on the current object.
setLong	Sets the value of a long field on the current object.
setShort	Sets the value of a short field on the current object.
toString	Overridden. Returns the string description of the current Field object.

Protected Methods

Name	Description
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

Field Class

Concepts

java.lang.reflect Package

Field Methods

Public Methods

Name	Description
equals	Overridden. Compares the current Field to a specified object.
get	Returns the value of a specific Field in the current object.
getBoolean	Retrieves the value of a static or instance field of the type boolean.
getByte	Retrieves the value of a static or instance field of the type byte.
getChar	Retrieves the value of a static or instance field of the type char or of another type that can be converted to char.
getDeclaringClass	Returns the name of the class that declares the current field object.
getDouble	Returns the value of a double field object or of another primitive type that can be converted to double.
getFloat	Returns the value of a float field object or of another primitive type that can be converted to float.
hashCode	Overridden. Returns the hash code of a Field object.
getInt	Returns the value of a int field object or of another primitive type that can be converted to int.
getLong	Returns the value of a long field object or of another primitive type that can be converted to long.
getModifiers	Returns an int that represents the modifiers for the field in the current Field object.
getName	Returns the name of the field of the current Field object.
getShort	Returns the value of a short field object or of another primitive type that can be converted to short.
getType	Returns the field type of the current field object.
getClass	(inherited from Object)
set	Sets the value of the current field of the specified object to a specified value.
setBoolean	Sets the value of a boolean field on the current object.
setByte	Sets the value of a byte field on the current object.
setChar	Sets the value of a char field on the current object.
setDouble	Sets the value of a double field on the current object.
setFloat	Sets the value of a float field on the current object.
setInt	Sets the value of an int field on the current object.

setLong	Sets the value of a long field on the current object.
setShort	Sets the value of a short field on the current object.
toString	Overridden. Returns the string description of the current Field object.

Protected Methods

Name	Description
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

Field Class

Concepts

java.lang.reflect Package

Field.equals Method

Compares the current Field to a specified object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

true if the two objects are identical; false otherwise.

Example

```
// Fld-eq1.jsl
// Field.Equals example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.IllegalAccessException,
        java.lang.IllegalArgumentException,
           java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class cls = java.awt.Point.class;

        // Declare and initialize a field object:
        Field field = cls.getField("x");

        // Declare and initialize two Point objects:
        Object p1 = new java.awt.Point(10,20);
        Object p2 = new java.awt.Point(10,30);

        // Declare and initialize two field objects:
        Object x1 = field.get(p1);
        Object x2 = field.get(p2);

        // Check if the two "x" fields are identical:
        System.out.println(x1.Equals(x2));
    }
}

/*
Output:
true
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.get Method

Returns the value of a specific Field in the current object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object get(
    java.lang.Object obj) throws java.lang.IllegalArgumentException, java.lang.IllegalAcces
sException;
```

Parameters

obj

The object that contains the specified field.

Return Value

The value of the specific Field in the current object.

Example

In this example, you retrieve the value of the field "x" on the current instance of the class **java.awt.Point**.

```
// Fld-get1.jsl
// Field.get example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.IllegalAccessException,
        java.lang.IllegalArgumentException,
           java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.awt.Point.class;

        // Declare and initialize a field object:
        Field field = mc.getField("x");

        // Declare and initialize a Point object:
        Object p1 = new java.awt.Point(10,20);

        // Display the value of the field represented by "field":
        System.out.println("The value of the 'x' field is: " +
                           field.get(p1));
    }
}

/*
Output:
The value of the 'x' field is: 10
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getBoolean Method

Retrieves the value of a static or instance field of the type boolean.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public boolean getBoolean(
    java.lang.Object obj) throws java.lang.IllegalArgumentException, java.lang.IllegalAcces
sException;
```

Parameters

obj

The object that contains the boolean value.

Return Value

The boolean value of the field.

Example

See the example on getByte.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getByte Method

Retrieves the value of a static or instance field of the type byte.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public byte getByte(
    java.lang.Object obj) throws java.lang.IllegalArgumentException, java.lang.IllegalAcces
sException;
```

Parameters

obj

The object that contains the byte value.

Return Value

The byte value of the field.

Example

```
// Fld-getBy1.jsl
// Field.getByte example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.IllegalAccessException,
        java.lang.IllegalArgumentException,
           java.lang.NoSuchFieldException
    {
        // Declare a Class object:
        Class mc = java.lang.Byte.class;

        // Get a field:
        Field field = mc.getField("MIN_VALUE");

        // Declare a Point object:
        Byte p1 = new Byte((byte)88);

        // Display the value of MIN_VALUE represented by "field":
        System.out.println("The value of the 'MIN_VALUE' field is: " +
                           field.getByte(p1));
    }
}

/*
Output:
The value of the 'MIN_VALUE' field is: -128
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getChar Method

Retrieves the value of a static or instance field of the type char or of another type that can be converted to char.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public char getChar(
    java.lang.Object obj) throws java.lang.IllegalArgumentException, java.lang.IllegalAcces
sException;
```

Parameters

obj

The object that contains the char value.

Return Value

The char value of the field.

Example

See the example on getByte.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getDeclaringClass Method

Returns the name of the class that declares the current field object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class getDeclaringClass();
```

Return Value

The name of the class that declares the current field object.

Example

```
// Fld-getDclass1.jsl
// Field.getDeclaringClass example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.NoSuchFieldException,
        java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.lang.Integer.class;

        // Declare and initialize a field object:
        Field field = mc.getField("MAX_VALUE");

        // Get and display the declaring class name:
        Class dc = field.getDeclaringClass();
        System.out.println("The declaring class is: " + dc);
    }
}

/*
Output:
The declaring class is: class java.lang.Integer
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getDouble Method

Returns the value of a double field object or of another primitive type that can be converted to double.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public double getDouble(
    java.lang.Object obj) throws java.lang.IllegalArgumentException, java.lang.IllegalAcces
sException;
```

Parameters

obj

The object that contains the double value.

Return Value

The double value of the field.

Example

See the example on getByte.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getFloat Method

Returns the value of a float field object or of another primitive type that can be converted to float.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public float getFloat(
    java.lang.Object obj) throws java.lang.IllegalArgumentException, java.lang.IllegalAcces
sException;
```

Parameters

obj

The object that contains the float value.

Return Value

The float value of the field.

Example

See the example on getByte.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.hashCode Method

Returns the hash code of a Field object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code of the Field object.

Example

```
// Fld-getHcode1.jsl
// Field.GetHashCode example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.IllegalAccessException,
        java.lang.IllegalArgumentException,
           java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.lang.Byte.class;

        // Declare and initialize a field object:
        Field field = mc.getField("MIN_VALUE");

         // Display hash code of the field:
        System.out.println("The hash code of the field is: " +
                           field.GetHashCode());
    }
}

/*
Output:
The hash code of the field is: 1704215444
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getInt Method

Returns the value of a int field object or of another primitive type that can be converted to int.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public int getInt(
    java.lang.Object obj) throws java.lang.IllegalArgumentException, java.lang.IllegalAcces
sException;
```

Parameters

obj

The object that contains the int value

Return Value

The int value of the field.

Example

See the example on getByte.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getLong Method

Returns the value of a long field object or of another primitive type that can be converted to long.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public long getLong(
    java.lang.Object obj) throws java.lang.IllegalArgumentException, java.lang.IllegalAcces
sException;
```

Parameters

obj

The object that contains the long value.

Return Value

The long value of the field.

Example

See the example on getByte.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getModifiers Method

Returns an int that represents the modifiers for the field in the current Field object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public int getModifiers();
```

Return Value

The modifiers for the field in the current Field object. This is a bitset representing a combination of modifiers defined in the Modifier class.

Example

```
// Fld-getmod1.jsl
// Field.getModifiers example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.IllegalAccessException,
        java.lang.IllegalArgumentException,
           java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.lang.Byte.class;

        // Declare and initialize a field object:
        Field field = mc.getField("MIN_VALUE");

        // Get the integer value of the modifiers:
        int mod = field.getModifiers();

        // Get the string value of the modifiers:
        System.out.println("The modifiers are: " +
                           Modifier.toString(mod));
    }
}

/*
Output:
The modifiers are: public static final
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getName Method

Returns the name of the field of the current Field object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getName();
```

Return Value

The name of the field of the current Field object.

Example

```
// Fld-getName1.jsl
// Field.getName example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.NoSuchFieldException ,
             java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.lang.Integer.class;

        // Declare and initialize a field object:
        Field field = mc.getField("MAX_VALUE");

        // Get and display the field's name:
        String name = field.getName();
        System.out.println("The field name is: " + field.getName());
    }
}

/*
Output:
The field name is: MAX_VALUE
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getShort Method

Returns the value of a short field object or of another primitive type that can be converted to short.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public short getShort(
    java.lang.Object obj) throws java.lang.IllegalArgumentException, java.lang.IllegalAcces
sException;
```

Parameters

obj

The object that contains the short value.

Return Value

The short value of the field.

Example

See the example on getByte.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.getType Method

Returns the field type of the current field object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class getType();
```

Return Value

The field type of the current field object.

Example

```
// Fld-getTyp1.jsl
// Field.getType example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.NoSuchFieldException,
        java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.lang.Integer.class;

        // Declare and initialize a field object:
        Field field = mc.getField("MAX_VALUE");

        // Get and display the type of the field:
        Class ft = field.getType();
        System.out.println("The type of the field is: " + ft);
    }
}

/*
Output:
The type of the field is: int
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.set Method

Sets the value of the current field of the specified object to a specified value.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void set(
    java.lang.Object obj,
    java.lang.Object value) throws java.lang.IllegalArgumentException, java.lang.IllegalAcc
essException;
```

Parameters

obj

The object that contains the field to be modified.

value

The new value of the field to be set.

Example

```
// Fld-set1.jsl
// Field.set example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.IllegalAccessException,
        java.lang.IllegalArgumentException,
           java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.awt.Point.class;

        // Declare and initialize a field object:
        Field field = mc.getField("x");

        // Declare and initialize a Point object:
        Object p1 = new java.awt.Point(10,20);

        // Display the value of the field represented by "field":
        System.out.println("The current value of the 'x' field is: " +
                                            field.get(p1));
        // Set the value of the field to 25:
        field.set(p1,new Integer(25));

        // Display the new value of the field represented by "field":
        System.out.println("The new value of the 'x' field is: " +
                                            field.get(p1));
    }
}

/*
Output:
The current value of the 'x' field is: 10
The new value of the 'x' field is: 25
*/
```

See Also

Reference

Field Class
Concepts
Field Members
java.lang.reflect Package

Field.setBoolean Method

Sets the value of a boolean field on the current object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void setBoolean(
    java.lang.Object obj,
    boolean z) throws java.lang.IllegalArgumentException, java.lang.IllegalAccessException;
```

Parameters

obj

The object that contains the field to be set.

z

The new value of the field.

Example

See the example on setInt.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.setByte Method

Sets the value of a byte field on the current object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void setByte(
    java.lang.Object obj,
    byte b) throws java.lang.IllegalArgumentException, java.lang.IllegalAccessException;
```

Parameters

obj

The object that contains the field to be set.

b

The new value of the field.

Example

See the example on setInt.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.setChar Method

Sets the value of a char field on the current object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void setChar(
    java.lang.Object obj,
    char c) throws java.lang.IllegalArgumentException, java.lang.IllegalAccessException;
```

Parameters

obj

The object that contains the field to be set.

c

The new value of the field.

Example

See the example on setInt.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.setDouble Method

Sets the value of a double field on the current object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void setDouble(
    java.lang.Object obj,
    double d) throws java.lang.IllegalArgumentException, java.lang.IllegalAccessException;
```

Parameters

obj

The object that contains the field to be set.

d

The new value of the field.

Example

See the example on setInt.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.setFloat Method

Sets the value of a float field on the current object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void setFloat(
    java.lang.Object obj,
    float f) throws java.lang.IllegalArgumentException, java.lang.IllegalAccessException;
```

Parameters

obj

The object that contains the field to be set.

f

The new value of the field.

Example

See the example on setInt.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.setInt Method

Sets the value of an int field on the current object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void setInt(
    java.lang.Object obj,
    int i) throws java.lang.IllegalArgumentException, java.lang.IllegalAccessException;
```

Parameters

obj

The object that contains the field to be set.

i

The new value of the field.

Example

```
// Fld-setInt1.jsl
// Field.setInt example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.IllegalAccessException,
        java.lang.IllegalArgumentException,
           java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.awt.Point.class;

        // Declare and initialize a field object:
        Field field = mc.getField("x");

        // Declare and initialize a Point object:
        Object p1 = new java.awt.Point(10,20);

        // Display the value of the field represented by "field":
        System.out.println("The current value of the 'x' field is: " +
                                            field.get(p1));
        // Set the value of the field to 22:
        field.setInt(p1,22);

        // Display the new value of the field represented by "field":
        System.out.println("The new value of the 'x' field is: " +
                                            field.get(p1));
    }
}

/*
Output:
The current value of the 'x' field is: 10
The new value of the 'x' field is: 22
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.setLong Method

Sets the value of a long field on the current object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void setLong(
    java.lang.Object obj,
    long l) throws java.lang.IllegalArgumentException, java.lang.IllegalAccessException;
```

Parameters

obj

The object that contains the field to be set.

l

The new value of the field.

Example

See the example on setInt.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.setShort Method

Sets the value of a short field on the current object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void setShort(
    java.lang.Object obj,
    short s) throws java.lang.IllegalArgumentException, java.lang.IllegalAccessException;
```

Parameters

obj

The object that contains the field to be set.

s

The new value of the field.

Example

See the example on setInt.

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

Field.toString Method

Returns the string description of the current Field object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The string description of the current Field object.

Example

```
// Fld-toStr1.jsl
// Field.toString example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.NoSuchFieldException,
        java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.lang.Integer.class;

        // Declare and initialize a field object:
        Field field = mc.getField("MAX_VALUE");

        // Display the description of the field:
        System.out.println(
              "The description of the field is:\n" +
              field.toString());
    }
}

/*
Output:
The description of the field is:
public static final int java.lang.Integer.MAX_VALUE
*/
```

See Also

Reference

Field Class

Concepts

Field Members

java.lang.reflect Package

InvocationTargetException Class

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.reflect.InvocationTargetException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.reflect.InvocationTargetException

See Also

Concepts

InvocationTargetException Members

java.lang.reflect Package

InvocationTargetException Members

The following tables list the members exposed by the InvocationTargetException type.

Public Constructors

Name	Description
InvocationTargetException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden.
getTargetException	
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InvocationTargetException Class

Concepts

java.lang.reflect Package

InvocationTargetException Constructor

Overload List

Name	Description
InvocationTargetException ()	
InvocationTargetException (Throwable)	
InvocationTargetException (SerializationInfo, StreamingContext)	
InvocationTargetException (String, Exception)	
InvocationTargetException (Throwable, String)	

See Also

Reference

InvocationTargetException Class

Concepts

InvocationTargetException Members

java.lang.reflect Package

InvocationTargetException Constructor ()

Initializes a new instance of the InvocationTargetException Class .

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.reflect.InvocationTargetException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

InvocationTargetException Class

Concepts

InvocationTargetException Members

java.lang.reflect Package

InvocationTargetException Constructor (Throwable)

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.reflect.InvocationTargetException(
    java.lang.Throwable exc);
```

Parameters

exc

See Also

Reference

InvocationTargetException Class

Concepts

InvocationTargetException Members

java.lang.reflect Package

InvocationTargetException Constructor (SerializationInfo, StreamingContext)

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.reflect.InvocationTargetException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

InvocationTargetException Class

Concepts

InvocationTargetException Members

java.lang.reflect Package

InvocationTargetException Constructor (String, Exception)

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.reflect.InvocationTargetException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also
Reference
InvocationTargetException Class
Concepts
InvocationTargetException Members
java.lang.reflect Package

InvocationTargetException Constructor (Throwable, String)

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.reflect.InvocationTargetException(
    java.lang.Throwable exc,
    java.lang.String msg);
```

Parameters

exc

msg

See Also

Reference

InvocationTargetException Class

Concepts

InvocationTargetException Members

java.lang.reflect Package

InvocationTargetException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden.
getTargetException	
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

InvocationTargetException Class

Concepts

java.lang.reflect Package

InvocationTargetException.GetObjectData Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

InvocationTargetException Class

Concepts

InvocationTargetException Members

java.lang.reflect Package

InvocationTargetException.getTargetException Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Throwable getTargetException();
```

See Also

Reference

InvocationTargetException Class

Concepts

InvocationTargetException Members

java.lang.reflect Package

InvocationTargetException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

InvocationTargetException Class

Concepts

java.lang.reflect Package

Member Interface

An interface that contains methods to retrieve information about members, such as fields or methods, and constructors.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public interface java.lang.reflect.Member
```

See Also

Concepts

Member Members

java.lang.reflect Package

Member Members

An interface that contains methods to retrieve information about members, such as fields or methods, and constructors.

The following tables list the members exposed by the Member type.

Public Fields

Name	Description
DECLARED	A constant that represents declared members such as fields, constructors, and methods.
PUBLIC	A constant that represents all public members such as fields, constructors, and methods.

Public Methods

Name	Description
getDeclaringClass	Returns the declaring class of the current Member object.
getModifiers	Returns an int that represents the modifiers for the member or constructor in the current Member object.
getName	Returns the name of the current Member object.

See Also

Reference

Member Interface

Concepts

java.lang.reflect Package

Member Fields

Public Fields

Name	Description
DECLARED	A constant that represents declared members such as fields, constructors, and methods.
PUBLIC	A constant that represents all public members such as fields, constructors, and methods.

See Also

Reference

Member Interface

Concepts

java.lang.reflect Package

Member.DECLARED Field

A constant that represents declared members such as fields, constructors, and methods.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int DECLARED;
```

Example

```
// Mbr-dec1.jsl
// Member.DECLARED example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Get and display the Member fields:
        int d1 = Member.DECLARED;
        int d2 = Member.PUBLIC;
        System.out.println(d1+"\n"+ d2);
    }
}

/*
Output:
1
0
*/
```

See Also

Reference

Member Interface

Concepts

Member Members

java.lang.reflect Package

Member.PUBLIC Field

A constant that represents all public members such as fields, constructors, and methods.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int PUBLIC;
```

Example

See the example on DECLARED.

See Also

Reference

Member Interface

Concepts

Member Members

java.lang.reflect Package

Member Methods

Public Methods

Name	Description
getDeclaringClass	Returns the declaring class of the current Member object.
getModifiers	Returns an int that represents the modifiers for the member or constructor in the current Member object.
getName	Returns the name of the current Member object.

See Also

Reference

Member Interface

Concepts

java.lang.reflect Package

Member.getDeclaringClass Method

Returns the declaring class of the current Member object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Class getDeclaringClass();
```

Return Value

The declaring class of the current Member object.

Example

```
// Mbr-getdc1.jsl
// Member.getDeclaringClass example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.NoSuchFieldException,
        java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.lang.Integer.class;

        // Declare and initialize a field object:
        Field field = mc.getField("MAX_VALUE");

        // Get and display the field's declaring class:
        Class dc = field.getDeclaringClass();
        System.out.println("The declaring class is:\n" + dc);
    }
}

/*
Output:
The declaring class is:
class java.lang.Integer
*/
```

See Also

Reference

Member Interface

Concepts

Member Members

java.lang.reflect Package

Member.getModifiers Method

Returns an int that represents the modifiers for the member or constructor in the current Member object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public abstract int getModifiers();
```

Return Value

The modifiers for the member or constructor in the current Member object.

Example

```
// Mbr-getmod1.jsl
// Member.getModifiers example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.NoSuchFieldException,
        java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.lang.Integer.class;

        // Declare and initialize a field object:
        Field field = mc.getField("MAX_VALUE");

        // Display the field:
        System.out.println("The field is:\n" + field);

        // Get the value of the modifiers:
        int mod = field.getModifiers();

        // Display the description of the modifiers:
        System.out.println(
              "The modifiers of the field are:\n" +
               Modifier.toString(mod));
    }
}

/*
Output:
The field is:
public static final int java.lang.Integer.MAX_VALUE
The modifiers of the field are:
public static final
*/
```

See Also

Reference

Member Interface

Concepts

Member Members

java.lang.reflect Package

Member.getName Method

Returns the name of the current Member object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getName();
```

Return Value

The name of the current Member object.

Example

```
// Mbr-getname1.jsl
// Member.getName example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
      throws java.lang.NoSuchFieldException,
        java.lang.NoSuchFieldException
    {
        // Declare and initialize a Class object:
        Class mc = java.lang.Integer.class;

        // Declare and initialize a field object:
        Field field = mc.getField("MAX_VALUE");

        // Get and display the field's name:
        String name = mc.getName()+"."+field.getName();
        System.out.println("The name of the field is:\n" + name);
    }
}

/*
Output:
The name of the field is:
java.lang.Integer.MAX_VALUE
*/
```

See Also

Reference

Member Interface

Concepts

Member Members

java.lang.reflect Package

Method Class

Contains methods to obtain reflection information on method names, parameters, modifiers, and return values. It also provides a way to dynamically invoke methods.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public final class java.lang.reflect.Method
    extends java.lang.Object
    implements java.lang.reflect.Member
```

Inheritance Hierarchy

java.lang.Object

java.lang.reflect.Method

See Also

Concepts

Method Members

java.lang.reflect Package

Method Members

Contains methods to obtain reflection information on method names, parameters, modifiers, and return values. It also provides a way to dynamically invoke methods.

The following tables list the members exposed by the Method type.

Public Methods

Name	Description
equals	Overridden. Checks the equality between the current Method object and a specified object.
getDeclaringClass	Gets the class in which this method is declared.
getExceptionTypes	Returns a class object array that contains the exceptions to be thrown by the current Method object.
hashCode	Overridden. Retrieves the hash code for the current Method object.
getModifiers	Returns the modifiers defined on the current Method object.
getName	Returns the method name of the current Method object.
getParameterTypes	Returns the parameter types of a Method object.
getReturnType	Returns the return type for a Method object.
getClass	(inherited from Object)
invoke	Invokes the current Method object on the specified object with the specified parameters.
clone	
toString	Overridden. Returns the string representation of a Method object.

See Also

Reference

Method Class

Concepts

java.lang.reflect Package

Method Methods

Public Methods

Name	Description
equals	Overridden. Checks the equality between the current Method object and a specified object.
getDeclaringClass	Gets the class in which this method is declared.
getExceptionTypes	Returns a class object array that contains the exceptions to be thrown by the current Method object.
hashCode	Overridden. Retrieves the hash code for the current Method object.
getModifiers	Returns the modifiers defined on the current Method object.
getName	Returns the method name of the current Method object.
getParameterTypes	Returns the parameter types of a Method object.
getReturnType	Returns the return type for a Method object.
getClass	(inherited from Object)
invoke	Invokes the current Method object on the specified object with the specified parameters.
clone	
toString	Overridden. Returns the string representation of a Method object.

See Also

Reference

Method Class

Concepts

java.lang.reflect Package

Method.equals Method

Checks the equality between the current Method object and a specified object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

true if the two objects are equal; false otherwise.

Example

```
// meth-eq1.jsl
// Method.Equals example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare a Class object:
        Class mc = MyClass.class;

        // Declare the an array of methods:
        Method[] methArr = mc.getMethods();

       // Get the parameter types:
       System.out.println("The method is:");
       for (int i=0; i<methArr.length; i++)
       {
           if("sample".Equals(methArr[i].getName()))
               System.out.println(methArr[i]);
        }
    }

    public void sample (byte b, short s, int i, float f, double d)
    { }
}

/*
Output:
The method is:
public void MyClass.sample(byte,short,int,float,double)
*/
```

See Also

Reference

Method Class

Concepts

Method Members

java.lang.reflect Package

Method.getDeclaringClass Method

Gets the class in which this method is declared.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class getDeclaringClass();
```

Return Value

The name of the class in which the method is declared.

Example

In this example, you retrieve and display the declaring class for the method "sample," which is declared in the same class (MyClass).

```
// meth-getDecCl1.jsl
// Method.getDeclaringClass example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare a Class object:
        Class mc = MyClass.class;

        // Declare the an array of methods:
        Method[] methArr = mc.getMethods();

       // Get the declaring class:
       System.out.println("The declaring class is:");
       for (int i=0; i<methArr.length; i++)
       {
           // Get the declaring class for sample() only:
           if("sample".Equals(methArr[i].getName()))
               System.out.println(methArr[i].getDeclaringClass());
        }
    }

    // Declare a "sample" method just for testing:
    public void sample (byte b, short s, int i, float f, double d)
    { }
}

/*
Output:
The declaring class is:
class MyClass
*/
```

See Also

Reference

Method Class

Concepts

Method Members

java.lang.reflect Package

Method.getExceptionTypes Method

Returns a class object array that contains the exceptions to be thrown by the current Method object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class[] getExceptionTypes();
```

Return Value

The exceptions to be thrown by the current Method object.

Example

In this example, you list all the exceptions associated with the methods in the class java.io.BufferedInputStream. You can also display the method names by removing the comment from the line indicated in the code.

```
// meth-getEx1.jsl
// Method.getExceptionTypes example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare a Class object:
        Class mc = java.io.BufferedInputStream.class;

        // Declare the an array of methods:
        Method[] methArr = mc.getMethods();

       // Display methods and exceptions:
       for (int i=0; i<methArr.length; i++)
       {
           // Uncomment the following line to get method names as well:
           // System.out.println("Method: " + methArr[i]);
           Class[] ex = methArr[i].getExceptionTypes();
           for (int j=0; j<ex.length; j++)
               System.out.println("Exception: " + ex[j]);
        }
    }
}

/*
Output:
Exception: class java.lang.InterruptedException
Exception: class java.lang.InterruptedException
Exception: class java.lang.InterruptedException
Exception: class java.io.IOException
Exception: class java.io.IOException
Exception: class java.io.IOException
Exception: class java.io.IOException
Exception: class java.io.IOException
Exception: class java.io.IOException
Exception: class java.io.IOException
*/
```

See Also

Reference

Method Class

Concepts

Method Members
java.lang.reflect Package

Method.hashCode Method

Retrieves the hash code for the current Method object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code for the current Method object.

Example

This example lists the hash code values for all the methods on the class java.io.BufferedInputStream.

```
// meth-getHCode1.jsl
// Method.GetHashCode example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare a Class object:
        Class mc = java.io.BufferedInputStream.class;

        // Declare the an array of methods:
        Method[] methArr = mc.getMethods();

       // Display hash code for the methods:
       for (int i=0; i<methArr.length; i++)
           System.out.println(methArr[i].GetHashCode());
    }
}

/*
Output:
1434567118
1555087625
1818432310
-1886973443
-1886973443
-1886973443
676000111
-1861917916
-1707723623
185017471
-1707465506
-1707465506
-1754479707
-1707389895
1505445493
678854254
564763867
523916199
*/
```

See Also

Reference

Method Class

Concepts
Method Members
java.lang.reflect Package

Method.getModifiers Method

Returns the modifiers defined on the current Method object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public int getModifiers();
```

Return Value

The modifiers of the current Method object. Modifiers are returned as an integer that is a combination of bits defined in the Modifier class.

Example

This example lists the modifiers of all the methods on the class java.io.BufferedInputStream.

```
// meth-getMod1.jsl
// Method.getModifiers example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare a Class object:
        Class mc = java.io.BufferedInputStream.class;

        // Declare the method array:
        Method[] methArr = mc.getMethods();

        // Get the modifiers of the methods:
        System.out.println("The modifiers are:");
        for (int i=0; i<methArr.length; i++)
        {
            // Get the modifier integer:
            int mod = methArr[i].getModifiers();
            // Convert the integer to a description string:
            System.out.println(Modifier.toString(mod));
        }
    }
}

/*
Output:
public final
public final
public final
public final
public final
public final
public synchronized
public synchronized
public synchronized
public
public synchronized
public synchronized
public synchronized
public synchronized
public
public
```

```
  public
  public
  */
```

See Also

Reference

Method Class

Concepts

Method Members

java.lang.reflect Package

Method.getName Method

Returns the method name of the current Method object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getName();
```

Return Value

The method name of the current Method object.

Example

This example lists the method names of the methods on the class java.io.BufferedInputStream.

```
// meth-getName.jsl
// Method.getName example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare a Class object:
        Class cls = MyClass.class;

        // Get declared methods:
        Method[] meth = cls.getDeclaredMethods();

        // Display the names of the declared methods:
        System.out.println("The declared methods are:");
        for (int i=0; i<meth.length; i++)
            System.out.println(meth[i].getName());
    }
}

/*
Output:
The declared methods are:
main
*/
```

See Also

Reference

Method Class

Concepts

Method Members

java.lang.reflect Package

Method.getParameterTypes Method

Returns the parameter types of a Method object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class[] getParameterTypes();
```

Return Value

The parameter types of the Method object.

Example

```
// meth-getparTyp1.jsl
// Method.getParameterTypes example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare a Class object:
        Class mc = MyClass.class;

        // Declare the an array of methods:
        Method[] methArr = mc.getMethods();

       // Get the parameter types:
       for (int i=0; i<methArr.length; i++)
       {
            if(!"sample".equals(methArr[i].getName()))
                continue;
            System.out.println(methArr[i]);
            System.out.println("The parameter types are:");

            // Declare the array of parameter types:
            Class[] paramTyArr = methArr[i].getParameterTypes();
                for (int j=0; j<paramTyArr.length; j++)
                     System.out.println("  " + j +" " +paramTyArr[j]);
        }
    }

    // Declare the "sample" method - just for testing:
    public void sample
     (byte b, short s, int i, float f, double d, String str, byte[] ba)
    { }
}

/*
Output:
public void MyClass.sample(byte,short,int,float,double,java.lang.String,byte[])
The parameter types are:
  0 byte
  1 short
  2 int
  3 float
  4 double
  5 class java.lang.String
  6 class [B*/
```

See Also

Reference

Method Class

Concepts

Method Members

java.lang.reflect Package

Method.getReturnType Method

Returns the return type for a Method object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Class getReturnType();
```

Return Value

The return type for the Method object.

Example

The following example returns the return types for the method "test" declared in the same class.

```
// meth-getretTy1.jsl
// Method.getReturnType example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare and initialize a Class object:
        Class mc = MyClass.class;

        // Declare the an array of methods:
        Method[] methArr = mc.getMethods();

       // Display return types for the methods:
       System.out.println("The return type(s):");
       for (int i=0; i<methArr.length; i++)
       {
           if(!"test".equals(methArr[i].getName()))
                 continue;
           System.out.println(methArr[i].getReturnType());
       }
    }

    // Declare a test method - just for testing:
    public void test(byte b, short s, int i, float f, double d) { }
}

/*
Output:
The return type(s):
void
*/
```

See Also

Reference

Method Class

Concepts

Method Members

java.lang.reflect Package

Method.invoke Method

Invokes the current Method object on the specified object with the specified parameters.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object invoke(
    java.lang.Object obj,
    java.lang.Object[] args) throws java.lang.IllegalAccessException, java.lang.IllegalArgu
mentException, java.lang.reflect.InvocationTargetException;
```

Parameters

obj

The object from which the method is invoked.

args

The arguments used to invoke the method.

Return Value

The object that represents the result of invoking the method.

Example

```
// meth-invo1.jsl
// Method.invoke example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
         throws java.lang.IllegalAccessException,
                java.lang.IllegalArgumentException,
                java.lang.reflect.InvocationTargetException,
                java.lang.NoSuchMethodException
    {
        // Declare and initialize a Class object:
        Class mc = MyClass.class;

        // Declare a method object:
        /* Using getMethod will return a method object for an accessibl
        public method. Note that the second param to getMethod() is the
        set of non-null Class objects, one for each param on the
        method. Also note that the specified parameter types must
        exactly match.*/

        Method meth = mc.getMethod("test", new Class[] {int.class});

        // Invoke the method "test":
        /* Note that the second parameter to invoke() is the set of
        arguments to be passed as params to the method that is invoked.
        If the parameter is a primitive type, the corresponding value
        in the argument array must be the appropriate wrapper object.*/

        Integer i = new Integer("50");
        System.out.println(meth.invoke(mc, new Object[] {i}));
    }

    // Declare "test" method - just for testing:
    public static int test(int i)
    {
```

```
            return 5*i;
        }
    }

    /*
    Output:
    250
    */
```

See Also

Reference

Method Class

Concepts

Method Members

java.lang.reflect Package

Method.toString Method

Returns the string representation of a Method object.

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

Return Value

The string representation of the Method object.

Example

```
// meth-ToString.jsl
// Method.ToString example

import java.lang.reflect.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Declare a Class object:
        Class mc = MyClass.class;

        // Declare the an array of methods:
        Method[] methArr = mc.getMethods();

       // Display the method "test":
       System.out.println("The methods of MyClass are:");
       for (int i=0; i<methArr.length; i++)
       {
           if("test".equals(methArr[i].getName()))
               System.out.println(methArr[i].ToString()) ;
       }
    }

    // Declare a test method - just for testing:
    public static void test(int x, String s) { }
}

/*
Output:
The methods of MyClass are:
public static void MyClass.test(int,java.lang.String)
*/
```

See Also

Reference

Method Class

Concepts

Method Members

java.lang.reflect Package

Modifier Class

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public class java.lang.reflect.Modifier
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.lang.reflect.Modifier

See Also

Concepts

Modifier Members

java.lang.reflect Package

Modifier Members

The following tables list the members exposed by the Modifier type.

Public Constructors

Name	Description
Modifier	

Public Fields

Name	Description
ABSTRACT	
FINAL	
INTERFACE	
NATIVE	
PRIVATE	
PROTECTED	
PUBLIC	
STATIC	
SYNCHRONIZED	
TRANSIENT	
VOLATILE	

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
isAbstract	
isFinal	
isInterface	
isNative	
isPrivate	

isProtected	
isPublic	
isStatic	
isSynchronized	
isTransient	
isVolatile	
clone	
toString	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Modifier Class

Concepts

java.lang.reflect Package

Modifier Fields

Public Fields

Name	Description
ABSTRACT	
FINAL	
INTERFACE	
NATIVE	
PRIVATE	
PROTECTED	
PUBLIC	
STATIC	
SYNCHRONIZED	
TRANSIENT	
VOLATILE	

See Also

Reference

Modifier Class

Concepts

java.lang.reflect Package

Modifier.ABSTRACT Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int ABSTRACT;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.FINAL Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int FINAL;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.INTERFACE Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int INTERFACE;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.NATIVE Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int NATIVE;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.PRIVATE Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int PRIVATE;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.PROTECTED Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int PROTECTED;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.PUBLIC Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int PUBLIC;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.STATIC Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int STATIC;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.SYNCHRONIZED Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int SYNCHRONIZED;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.TRANSIENT Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int TRANSIENT;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.VOLATILE Field

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static final int VOLATILE;
```

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier Constructor

Initializes a new instance of the Modifier Class .

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public java.lang.reflect.Modifier();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier Methods

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
isAbstract	
isFinal	
isInterface	
isNative	
isPrivate	
isProtected	
isPublic	
isStatic	
isSynchronized	
isTransient	
isVolatile	
clone	
toString	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Modifier Class

Concepts

java.lang.reflect Package

Modifier.isAbstract Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isAbstract(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isFinal Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isFinal(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isInterface Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isInterface(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isNative Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isNative(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isPrivate Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isPrivate(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isProtected Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isProtected(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isPublic Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isPublic(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isStatic Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isStatic(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isSynchronized Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isSynchronized(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isTransient Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isTransient(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.isVolatile Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

```
public static boolean isVolatile(
    int mod);
```

Parameters

mod

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

Modifier.toString Method

Package: java.lang.reflect

Assembly: vjslib (in vjslib.dll)

See Also

Reference

Modifier Class

Concepts

Modifier Members

java.lang.reflect Package

java.lang.String

Represents an immutable sequence of Unicode characters.

J#

```
public final class String implements Serializable
```

Example

```
// string_overview.jsl

import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        // charAt
        // ============================================================
        String str1 = "Hello, World!";

        // Get the character at positions 0 and 12.
        int index1 = str1.charAt(0);
        int index2 = str1.charAt(12);

        // Print out the results.
        System.out.println("charAt");
        System.out.println("The character at position 0 is " +
            (char)index1);
        System.out.println("The character at position 12 is " +
            (char)index2);

        // concat
        // ============================================================
        String str2 = "This is the beginning ";
        String str3 = "of the new String.";

        // Concatenate the two strings together.
        String str4 = str2.concat(str3);

        // Display the new String.
        System.out.println("\nconcat");
        System.out.println("The concatenated string: " + str4);

        // copyValueOf
        // ============================================================
        // Populate a character array with data.
        char[] arr1 = new char[] {
            '1', '2', '3', '4'
        };

        // Create a String containig the contents of arr
        // starting at index 1 for length 2.
        String str5 = String.copyValueOf(arr1, 1, 2);

        // Display the results of the new String.
        System.out.println("\ncopyValueOf");
        System.out.println("The new String contains \"" + str5 +
            "\"");

        // endsWith
        // ============================================================
        String str6 = "My name is Fred";
        String str7 = "My name is Xiao Mei";
```

```
// The String to check the above two Strings to see
// if they end with this value (ei).
String endStr = "ei";

// Do either of the first two Strings end with endStr?
boolean ends1 = str6.endsWith(endStr);
boolean ends2 = str7.endsWith(endStr);

// Display the results of the endsWith calls.
System.out.println("\nendsWith");
System.out.println("\"" + str6 + "\" ends with " +
    "\"" + endStr + "\"? " + ends1);
System.out.println("\"" + str7 + "\" ends with " +
    "\"" + endStr + "\"? " + ends2);

// regionMatches
// ========================================================
String str8 = "Missouri Florida Colorado Washington";
String str9 = "Washington Colorado Florida Missouri";

// Determine whether characters 0 through 7 in str8
// match characters 28 through 35 in str9.
boolean match1 = str8.regionMatches(0, str9, 28, 8);

// Determine whether characters 9 through 15 in str8
// match characters 9 through 15 in str9.
boolean match2 = str8.regionMatches(9, str9, 9, 8);

// Display the results of the regionMatches method calls.
System.out.println("\nregionMatches");
System.out.println("str8[0 - 7] == str9[28 - 35]? " + match1);
System.out.println("str8[9 - 15] == str9[9 - 15]? " + match2);

// replace
// ========================================================
String str10 = "The quick brown fox jumps over the lazy dog.";

// Replace all the 'd' characters with 'l' characters.
String str11 = str10.replace('d', 'l');

// Display the strings for comparison.
System.out.println("\nreplace");
System.out.println("old = " + str10);
System.out.println("new = " + str11);

// substring
// ========================================================
String str12 = "Navy blue is my favorite color.";

// Get a substring of the above string starting from
// index 5.
String str13 = str12.substring(5);

// Display the two strings for comparison.
System.out.println("\nsubstring");
System.out.println("old = " + str12);
System.out.println("new = " + str13);

// toLowerCase
// ========================================================
String str14 = "ThIs Is HaRd To ReAd.";

// Convert the above string to all lowercase.
String lowerStr = str14.toLowerCase();

// Display the two strings for comparison.
System.out.println("\ntoLowerCase");
```

```
        System.out.println("old = " + str14);
        System.out.println("lowercase = " + lowerStr);

        // toUpperCase
        // ============================================================
        String str15 = "ThIs Is HaRd To ReAd.";

        // Convert the above string to all uppercase
        // using the default system Locale.
        Locale loc = Locale.getDefault();
        String upperStr = str15.toUpperCase(loc);

        // Display the two strings for comparison.
        System.out.println("\ntoUpperCase");
        System.out.println("old = " + str15);
        System.out.println("uppercase = " + upperStr);

        // valueOf
        // ============================================================
        String hundredStr = String.valueOf(100);
        String minStr = String.valueOf(Integer.MIN_VALUE);

        // Display the string representations of the above
        // int values.
        System.out.println("\nvalueOf");
        System.out.println(hundredStr);
        System.out.println(minStr);
    }
}

/*
Output:
charAt
The character at position 0 is H
The character at position 12 is !

concat
The concatenated string: This is the beginning of the new String.

copyValueOf
The new String contains "23"

endsWith
"My name is Fred" ends with "ei"? false
"My name is Xiao Mei" ends with "ei"? true

regionMatches
str8[0 - 7] == str9[28 - 35]? true
str8[9 - 15] == str9[9 - 15]? false

replace
old = The quick brown fox jumps over the lazy dog.
new = The quick brown fox jumps over the lazy log.

substring
old = Navy blue is my favorite color.
new = blue is my favorite color.

toLowerCase
old = ThIs Is HaRd To ReAd.
lowercase = this is hard to read.

toUpperCase
old = ThIs Is HaRd To ReAd.
uppercase = THIS IS HARD TO READ.

valueOf
100
```

```
-2147483648
*/
```

See Also

Concepts

Visual J# Class Library

java.lang.String.String()

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object.

J#

```
public String()
```

See Also

Reference

java.lang.String

java.lang.String.String(String src)

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of an existing **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object.

J#

```
public String(String src)
```

Parameters

Parameter	Description
src	An existing String whose contents will be used to populate the String.

See Also

Reference

java.lang.String

java.lang.String.String(StringBuffer buffer)

Initializes a new instance of a String object with the contents of a StringBuffer object.

J#

```
public String(StringBuffer buffer)
```

Parameters

Parameter	Description
buffer	A **StringBuffer** object whose contents will be used to populate the String.

See Also

Reference

java.lang.String

java.lang.String.String(char[] charArray)

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of a character array.

J#

```
public String(char[] charArray)
```

Parameters

Parameter	Description
charArray	A character array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

See Also

Reference

java.lang.String

java.lang.String.String(char[] charArray, int offset, int count)

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of a character array.

J#

```
public String(char[] charArray, int offset, int count)
```

Parameters

Parameter	Description
charArray	A character array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
offset	The first character in the character array to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. This value must be greater than zero and less than the length of the character array.
count	The number of characters within the character array, after the offset, to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. If this value is greater than the number of characters in the character array after the offset, then only the remaining characters are copied into the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

See Also

Reference

java.lang.String

java.lang.String.String(byte[] byteArray)

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of a byte array.

J#

```
public String(byte[] byteArray)
```

Parameters

Parameter	Description
byteArray	A byte array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

See Also

Reference

java.lang.String

java.lang.String.String(byte[] byteArray, int offset, int count)

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of a byte array.

J#

```
public String(byte[] byteArray, int offset, int count)
```

Parameters

Parameter	Description
byteArray	A byte array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
offset	The first byte in the byte array to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. This value must be greater than zero and less than the length of the byte array.
count	The number of bytes within the byte array, after the offset, to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. If this value is greater than the number of bytes in the byte array after the offset, then only the remaining bytes are copied into the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

See Also

Reference

java.lang.String

java.lang.String.String(byte[] byteArray, String enc)

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of a byte array using the given encoding.

J#

```
public String(byte[] byteArray, String enc) throws UnsupportedEncodingException
```

Parameters

Parameter	Description
byteArray	A byte array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
enc	The encoding for the newly created **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object.

See Also

Reference

java.lang.String

java.lang.String.String(byte[] byteArray, int offset, int count, String enc)

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of a byte array using the given encoding.

J#

```
public String(byte[] byteArray, int offset, int count, String enc) throws UnsupportedEncodi
ngException
```

Parameters

Parameter	Description
byteArray	A byte array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
offset	The first byte in the byte array to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. This value must be greater than zero and less than the length of the byte array.
count	The number of bytes within the byte array, after the offset, to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. If this value is greater than the number of bytes in the byte array after the offset, then only the remaining bytes are copied into the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
enc	The encoding for the newly created **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object.

See Also

Reference

java.lang.String

java.lang.String.String(byte[] asciiArray, int hibyte)

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of an ASCII array.

J#

```
public String(byte[] asciiArray, int hibyte)
```

Parameters

Parameter	Description
asciiArray	An ASCII array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
hibyte	Set to 0 if asciiArray contains ASCII characters.

See Also

Reference

java.lang.String

java.lang.String.String(byte[] asciiArray, int hibyte, int offset, int count)

Initializes a new instance of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of an ASCII array.

J#

```
public String(byte[] asciiArray, int hibyte, int offset, int count)
```

Parameters

Parameter	Description
asciiArray	An ASCII array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
hibyte	Set to 0 if asciiArray contains ASCII characters.
offset	The first ASCII character in the ASCII array to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. This value must be greater than zero and less than the length of the ASCII array.
count	The number of ASCII characters within the ASCII array, after the offset, to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. If this value is greater than the number of ASCII characters in the ASCII array after the offset, then only the remaining ASCII characters are copied into the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

See Also

Reference

java.lang.String

java.lang.String.charAt(int index)

Gets the character at the given index within the **String**.

J#

```
public char charAt(int index)
```

Parameters

Parameter	Description
index	The position within the **String** whose character is to be retrieved. A StringIndexOutofBoundsException will be thrown if the index is invalid.

Property Value/Return Value

The **char** at the given index within the String.

Example

```
// string_charat.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "Hello, World!";

        // Get the character at positions 0 and 12.
        int index1 = str.charAt(0);
        int index2 = str.charAt(12);

        // Print out the results.
        System.out.println("The character at position 0 is " +
            (char)index1);
        System.out.println("The character at position 12 is " +
            (char)index2);
    }
}

/*
Output:
The character at position 0 is H
The character at position 12 is !
*/
```

See Also

Reference

java.lang.String

java.lang.String.compareTo(String str)

Compares two fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 objects for equality.

J#

```
public int compareTo(String str)
```

Parameters

Parameter	Description
str	A String object to compare with the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object for equality.

Property Value/Return Value

Less than zero if the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object is less than *str*, zero if the two **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects are equal, or greater than zero if *str* is greater than the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object.

Example

```
// string_compareto.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str1 = "This is string 1";
        String str2 = "This is string 2";

        // Compare the two strings.
        int result = str1.compareTo(str2);

        // Display the results of the comparison.
        if (result < 0)
        {
            System.out.println("\"" + str1 + "\" +
                " is less than " +
                "\"" + str2 + "\"");
        }
        else if (result == 0)
        {
            System.out.println("\"" + str1 + "\" +
                " is equal to " +
                "\"" + str2 + "\"");
        }
        else // if (result > 0)
        {
            System.out.println("\"" + str1 + "\" +
                " is greater than " +
                "\"" + str2 + "\"");
        }
    }
}

/*
Output:
"This is string 1" is less than "This is string 2"
*/
```

See Also

Reference

java.lang.String

java.lang.String.concat(String str)

Concatenates two fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 objects.

J#

```
public String concat(String str)
```

Parameters

Parameter	Description
str	A **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to append to the end of the current String object.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** that is a concatenation of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** and *str*.

Example

```
// string_concat.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str1 = "This is the beginning ";
        String str2 = "of the new String.";

        // Concatenate the two strings together.
        String str3 = str1.concat(str2);

        // Display the new String.
        System.out.println("The concatenated string: " + str3);
    }
}

/*
Output:
The concatenated string: This is the beginning of the new String.
*/
```

See Also

Reference

java.lang.String

java.lang.String.copyValueOf(char[] data, int offset, int count)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of a character array.

J#

```
public static String copyValueOf(char[] data, int offset, int count)
```

Parameters

Parameter	Description
data	A character array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
offset	The first character in the character array to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. This value must be greater than zero and less than the length of the character array.
count	The number of characters within the character array, after the offset, to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. If this value is greater than the number of characters in the character array after the offset, then only the remaining characters are copied into the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A String object with the contents of the character array.

Example

```
// string_copyvalueof1.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Populate a character array with data.
        char[] arr = new char[] { '1', '2', '3', '4' };

        // Create a String containig the contents of arr
        // starting at index 1 for length 2.
        String str = String.copyValueOf(arr, 1, 2);

        // Display the results of the new String.
        System.out.println("The new String contains \"" + str +
            "\"");
    }
}

/*
Output:
The new String contains "23"
*/
```

See Also

Reference

java.lang.String

java.lang.String.copyValueOf(char[] data)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object with the contents of a character array.

J#

```
public static String copyValueOf(char[] data)
```

Parameters

Parameter	Description
data	A character array whose contents will be used to populate the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A String object with the contents of the character array.

Example

```
// string_copyvalueof2.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Populate a character array with data.
        char[] arr = new char[] { '1', '2', '3', '4' };

        // Create a String containig the contents of arr.
        String str = String.copyValueOf(arr);

        // Display the results of the new String.
        System.out.println("The new String contains \"" + str +
            "\"");
    }
}

/*
Output:
The new String contains "1234"
*/
```

See Also

Reference

java.lang.String

java.lang.String.endsWith(String str)

Determines whether the current fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 ends with the contents of another **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

C#

```
public boolean endsWith(String str)
```

Parameters

Parameter	Description
str	The **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** whose contents will be compared against the end of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

true if the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** ends with the contents of *str*; **false** otherwise.

Example

```
// string_endswith.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str1 = "My name is Fred";
        String str2 = "My name is Xiao Mei";

        // The String to check the above two Strings to see
        // if they end with this value (ei).
        String endStr = "ei";

        // Do either of the first two Strings end with endStr?
        boolean ends1 = str1.endsWith(endStr);
        boolean ends2 = str2.endsWith(endStr);

        // Display the results of the endsWith calls.
        System.out.println("\"" + str1 + "\" ends with " +
            "\"" + endStr + "\"? " + ends1);
        System.out.println("\"" + str2 + "\" ends with " +
            "\"" + endStr + "\"? " + ends2);
    }
}

/*
Output:
"My name is Fred" ends with "ei"? false
"My name is Xiao Mei" ends with "ei"? true
*/
```

See Also

Reference

java.lang.String

java.lang.String.equals(Object obj)

Determines whether two fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 objects contain the same data.

J#

```
public boolean equals(Object obj)
```

Parameters

Parameter	Description
obj	A String object to compare against the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for equality.

Property Value/Return Value

true if the two **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects contain the same data; **false** otherwise.

Example

```
// string_equals.jsl

public class Program
{
    public static void main(String[] args)
    {
        String physicist1 = "Albert Einstein";
        String physicist2 = "Max Planck";
        String physicist3 = "Albert Einstein";

        // Are any of the above Strings equal to one another?
        boolean equals1 = physicist1.equals(physicist2);
        boolean equals2 = physicist1.equals(physicist3);

        // Display the results of the equality checks.
        System.out.println("\"" + physicist1 + "\" equals \"" +
            physicist2 + "\"? " + equals1);
        System.out.println("\"" + physicist1 + "\" equals \"" +
            physicist3 + "\"? " + equals2);
    }
}

/*
Output:
"Albert Einstein" equals "Max Planck"? false
"Albert Einstein" equals "Albert Einstein"? true
*/
```

See Also

Reference

java.lang.String

java.lang.String.equalsIgnoreCase(String str)

Determines whether two fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 objects contain the same data, ignoring the case of the letters in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

J#

```
public boolean equalsIgnoreCase(String str)
```

Parameters

Parameter	Description
str	A String object to compare against the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for equality.

Property Value/Return Value

true if the two **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects contain either exactly the same data or if they differ only in case; **false** otherwise.

Example

```
// string_equalsignorecase.jsl

public class Program
{
    public static void main(String[] args)
    {
        String physicist1 = "Albert Einstein";
        String physicist2 = "Max Planck";
        String physicist3 = "albert einstein";

        // Are any of the above Strings equal to one another?
        boolean equals1 = physicist1.equalsIgnoreCase(physicist2);
        boolean equals2 = physicist1.equalsIgnoreCase(physicist3);

        // Display the results of the equality checks.
        System.out.println("\"" + physicist1 + "\" equals \"" +
            physicist2 + "\"? " + equals1);
        System.out.println("\"" + physicist1 + "\" equals \"" +
            physicist3 + "\"? " + equals2);
    }
}

/*
Output:
"Albert Einstein" equals "Max Planck"? false
"Albert Einstein" equals "albert einstein"? true
*/
```

See Also

Reference

java.lang.String

java.lang.String.getBytes()

Gets the contents of the fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 as a byte array.

J#

```
public byte[] getBytes()
```

Property Value/Return Value

The contents of the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** as a byte array.

Example

```
// string_getbytes1.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "This is a String.";

        // Copy the contents of the String to a byte array.
        byte[] arr = str.getBytes();

        // Create a new String using the contents of the byte array.
        String newStr = new String(arr);

        // Display the contents of the byte array.
        System.out.println("The new String equals \"" +
            newStr + "\"");
    }
}

/*
Output:
The new String equals "This is a String."
*/
```

See Also

Reference

java.lang.String

java.lang.String.getBytes(String enc)

Gets the contents of the fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 as a byte array using the given encoding.

J#

```
public byte[] getBytes(String enc) throws UnsupportedEncodingException
```

Parameters

Parameter	Description
enc	The encoding for the contents of the newly created byte array.

Property Value/Return Value

A byte array containing the contents of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Example

```
// string_getbytes2.jsl

import java.io.UnsupportedEncodingException;

public class Program
{
    public static void main(String[] args)
    {
        String str = "This is a String.";

        try
        {
            // Copy the contents of the String to a byte array
            // using the ASCII encoding.
            byte[] arr = str.getBytes("ASCII");

            // Create a new String using the contents of the byte array.
            String newStr = new String(arr);

            // Display the contents of the byte array.
            System.out.println("The new String equals \"" +
                newStr + "\"");
        }
        catch (UnsupportedEncodingException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
The new String equals "This is a String."
*/
```

See Also

Reference

java.lang.String

java.lang.String.getBytes(int srcOffset, int srcEnd, byte[] dst, int dstOffset)

Gets the contents of the fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 as a byte array.

J#

```
public void getBytes(int srcOffset, int srcEnd, byte[] dst, int dstOffset)
```

Parameters

Parameter	Description
srcOffset	The index of the first character in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to copy into the byte array. A 5d87a3c3-8577-4fe2-8616-c28146468df9 is thrown if this index is invalid.
srcEnd	The index one greater than the last character in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to copy into the byte array. A **5d87a3c3-8577-4fe2-8616-c28146468df9** is thrown if this index is invalid.
dst	A byte array that will contain the contents of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
dstOffset	The index within the byte array that will hold the first character copied from the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. An 997bf792-c774-4384-8045-19bacb0cb4ad is thrown if this index is invalid.

Example

```
// string_getbytes3.jsl

import java.io.UnsupportedEncodingException;

public class Program
{
    public static void main(String[] args)
    {
        String str = "This is a String.";

        // Copy the contents of the String to a byte array.
        // Only copy characters 5 through 10 from str.
        // Fill the source array starting at position 3.
        byte[] arr = new byte[] { ' ', ' ', ' ', ' ',
                                  ' ', ' ', ' ', ' ' };
        str.getBytes(5, 10, arr, 3);

        // Create a new String using the contents of the byte array.
        String newStr = new String(arr);

        // Display the contents of the byte array.
        System.out.println("The new String equals \"" +
            newStr + "\"");
    }
}

/*
Output:
The new String equals "   is a "
*/
```

See Also

Reference

java.lang.String

java.lang.String.getChars(int srcOffset, int srcEnd, char[] dst, int dstOffset)

Gets the contents of the fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 as a character array.

J#

```
public void getChars(int srcOffset, int srcEnd, char[] dst, int dstOffset)
```

Parameters

Parameter	Description
srcOffset	The index of the first character in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to copy into the character array. A 5d87a3c3-8577-4fe2-8616-c28146468df9 is thrown if this index is invalid.
srcEnd	The index of the last character in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to copy into the character array. A **5d87a3c3-8577-4fe2-8616-c28146468df9** is thrown if this index is invalid.
dst	A character array that will contain the contents of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
dstOffset	The index within the character array that will hold the first character copied from the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. An 997bf792-c774-4384-8045-19bacb0cb4ad is thrown if this index is invalid.

Example

```
// string_getchars.jsl

import java.io.UnsupportedEncodingException;

public class Program
{
    public static void main(String[] args)
    {
        String str = "This is a String.";

        // Copy the contents of the String to a byte array.
        // Only copy characters 5 through 10 from str.
        // Fill the source array starting at position 3.
        char[] arr = new char[] { ' ', ' ', ' ', ' ',
                                  ' ', ' ', ' ', ' ' };
        str.getChars(5, 10, arr, 3);

        // Display the contents of the byte array.
        System.out.println("The char array equals \"" +
            arr + "\"");
    }
}

/*
Output:
The char array equals "   is a "
*/
```

See Also

Reference

java.lang.String

java.lang.String.hashCode()

Generates a hash value that can be used to uniquely identify a particular fa4e6a2d-54fd-4a79-8103-2fae784c7bf4.

J#

```
public int hashCode()
```

Property Value/Return Value

A hash value that uniquely identifies a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Example

```
// string_hashcode.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "Hello, World!";

        // Get the hash code for the above string.
        int hash = str.hashCode();

        // Display the hash code.
        System.out.println("The hash for \"" + str +
            "\" is " + hash);
    }
}

/*
Output:
The hash for "Hello, World!" is 4052569
*/
```

See Also

Reference

java.lang.String

java.lang.String.indexOf(int ch, int offset)

Gets the first index of a character within a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 after an offset.

J#

```
public int indexOf(int ch, int offset)
```

Parameters

Parameter	Description
ch	The character to search the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for.
offset	The position within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to begin searching for the character. This value can not be greater than the length of the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

The index of the first occurrence of *ch* in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** after the offset, or -1 if the character does not exist within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** or if an invalid offset was provided.

Example

```
// string_indexof1.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "The quick brown fox jumps over the lazy dog.";

        // Get the index of all the characters of the alphabet
        // starting from the beginning of the String.
        int a = str.indexOf('a', 0);
        int b = str.indexOf('b', 0);
        int c = str.indexOf('c', 0);
        int d = str.indexOf('d', 0);
        int e = str.indexOf('e', 0);
        int f = str.indexOf('f', 0);
        int g = str.indexOf('g', 0);
        int h = str.indexOf('h', 0);
        int i = str.indexOf('i', 0);
        int j = str.indexOf('j', 0);
        int k = str.indexOf('k', 0);
        int l = str.indexOf('l', 0);
        int m = str.indexOf('m', 0);
        int n = str.indexOf('n', 0);
        int o = str.indexOf('o', 0);
        int p = str.indexOf('p', 0);
        int q = str.indexOf('q', 0);
        int r = str.indexOf('r', 0);
        int s = str.indexOf('s', 0);
        int t = str.indexOf('t', 0);
        int u = str.indexOf('u', 0);
        int v = str.indexOf('v', 0);
        int w = str.indexOf('w', 0);
        int x = str.indexOf('x', 0);
        int y = str.indexOf('y', 0);
        int z = str.indexOf('z', 0);

        // Display the results of all the indexOf method calls.
        System.out.println(" a  b c  d e  f  g h i  j");
        System.out.println("========================");
        System.out.println(a + " " + b + " " + c + " " + d + " " +
```

```
                            e + " " + f + " " + g + " " + h + " " +
                            i + " " + j + "\n");

        System.out.println("k  l  m  n  o  p q  r  s  t");
        System.out.println("==========================");
        System.out.println(k + " " + l + " " + m + " " + n + " " +
                            o + " " + p + " " + q + " " + r + " " +
                            s + " " + t + "\n");

        System.out.println("u  v  w  x  y  z");
        System.out.println("===============");
        System.out.println(u + " " + v + " " + w + " " + x + " " +
                            y + " " + z);
    }
}

/*
Output:
 a  b c  d e  f  g h i  j
========================
36 10 7 40 2 16 42 1 6 20

k  l  m  n  o  p q  r  s  t
==========================
8 35 22 14 12 23 4 11 24 31

u  v  w  x  y  z
===============
5 27 13 18 38 37
*/
```

See Also

Reference

java.lang.String

java.lang.String.indexOf(int ch)

Gets the first index of a character within a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4.

J#

```
public int indexOf(int ch)
```

Parameters

Parameter	Description
ch	The character to search the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for.

Property Value/Return Value

The index of the first occurrence of *ch* in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**, or -1 if the character does not exist within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Example

```
// string_indexof2.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "The quick brown fox jumps over the lazy dog.";

        // Get the index of all the characters of the alphabet
        // starting from the beginning of the String.
        int a = str.indexOf('a');
        int b = str.indexOf('b');
        int c = str.indexOf('c');
        int d = str.indexOf('d');
        int e = str.indexOf('e');
        int f = str.indexOf('f');
        int g = str.indexOf('g');
        int h = str.indexOf('h');
        int i = str.indexOf('i');
        int j = str.indexOf('j');
        int k = str.indexOf('k');
        int l = str.indexOf('l');
        int m = str.indexOf('m');
        int n = str.indexOf('n');
        int o = str.indexOf('o');
        int p = str.indexOf('p');
        int q = str.indexOf('q');
        int r = str.indexOf('r');
        int s = str.indexOf('s');
        int t = str.indexOf('t');
        int u = str.indexOf('u');
        int v = str.indexOf('v');
        int w = str.indexOf('w');
        int x = str.indexOf('x');
        int y = str.indexOf('y');
        int z = str.indexOf('z');

        // Display the results of all the indexOf method calls.
        System.out.println(" a  b c  d e  f  g h i  j");
        System.out.println("=========================");
        System.out.println(a + " " + b + " " + c + " " + d + " " +
                           e + " " + f + " " + g + " " + h + " " +
                           i + " " + j + "\n");

        System.out.println("k  l  m  n  o  p q  r  s  t");
```

```
        System.out.println("==========================");
        System.out.println(k + " " + l + " " + m + " " + n + " " +
                           o + " " + p + " " + q + " " + r + " " +
                           s + " " + t + "\n");

        System.out.println("u  v  w  x  y  z");
        System.out.println("===============");
        System.out.println(u + " " + v + " " + w + " " + x + " " +
                           y + " " + z);
    }
}

/*
Output:
 a  b c  d e  f  g h i  j
=========================
36 10 7 40 2 16 42 1 6 20

k  l  m  n  o  p q  r  s  t
==========================
8 35 22 14 12 23 4 11 24 31

u  v  w  x  y  z
===============
5 27 13 18 38 37
*/
```

See Also

Reference

java.lang.String

java.lang.String.indexOf(String str, int offset)

Gets the first index of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 within a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** after an offset.

J#

```
public int indexOf(String str, int offset)
```

Parameters

Parameter	Description
str	The **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to search the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for.
offset	The position within the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to begin searching for the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. This value cannot be greater than the length of the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

The index of the first occurrence of *str* in the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** after the offset, or -1 if *str* does not exist within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** or if an invalid offset was provided.

Example

```
// string_indexof3.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "The quick brown fox jumps over the lazy dog.";

        // Get the index of all the characters of the alphabet
        // starting from the beginning of the String.
        int a = str.indexOf("a", 0);
        int b = str.indexOf("b", 0);
        int c = str.indexOf("c", 0);
        int d = str.indexOf("d", 0);
        int e = str.indexOf("e", 0);
        int f = str.indexOf("f", 0);
        int g = str.indexOf("g", 0);
        int h = str.indexOf("h", 0);
        int i = str.indexOf("i", 0);
        int j = str.indexOf("j", 0);
        int k = str.indexOf("k", 0);
        int l = str.indexOf("l", 0);
        int m = str.indexOf("m", 0);
        int n = str.indexOf("n", 0);
        int o = str.indexOf("o", 0);
        int p = str.indexOf("p", 0);
        int q = str.indexOf("q", 0);
        int r = str.indexOf("r", 0);
        int s = str.indexOf("s", 0);
        int t = str.indexOf("t", 0);
        int u = str.indexOf("u", 0);
        int v = str.indexOf("v", 0);
        int w = str.indexOf("w", 0);
        int x = str.indexOf("x", 0);
        int y = str.indexOf("y", 0);
        int z = str.indexOf("z", 0);
```

```
        // Display the results of all the indexOf method calls.
        System.out.println(" a  b c  d e  f  g h i  j");
        System.out.println("========================");
        System.out.println(a + " " + b + " " + c + " " + d + " " +
                           e + " " + f + " " + g + " " + h + " " +
                           i + " " + j + "\n");

        System.out.println("k  l  m  n  o  p q  r  s  t");
        System.out.println("==========================");
        System.out.println(k + " " + l + " " + m + " " + n + " " +
                           o + " " + p + " " + q + " " + r + " " +
                           s + " " + t + "\n");

        System.out.println("u  v  w  x  y  z");
        System.out.println("===============");
        System.out.println(u + " " + v + " " + w + " " + x + " " +
                           y + " " + z);
    }
}

/*
Output:
 a  b c  d e  f  g h i  j
========================
36 10 7 40 2 16 42 1 6 20

k  l  m  n  o  p q  r  s  t
==========================
8 35 22 14 12 23 4 11 24 31

u  v  w  x  y  z
===============
5 27 13 18 38 37
*/
```

See Also

Reference

java.lang.String

java.lang.String.indexOf(String str)

Gets the first index of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 within a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

J#

```
public int indexOf(String str)
```

Parameters

Parameter	Description
str	The **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to search the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for.

Property Value/Return Value

The index of the first occurrence of *str* in the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**, or -1 if *str* does not exist within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Example

```
// string_indexof4.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "The quick brown fox jumps over the lazy dog.";

        // Get the index of all the characters of the alphabet
        // starting from the beginning of the String.
        int a = str.indexOf("a");
        int b = str.indexOf("b");
        int c = str.indexOf("c");
        int d = str.indexOf("d");
        int e = str.indexOf("e");
        int f = str.indexOf("f");
        int g = str.indexOf("g");
        int h = str.indexOf("h");
        int i = str.indexOf("i");
        int j = str.indexOf("j");
        int k = str.indexOf("k");
        int l = str.indexOf("l");
        int m = str.indexOf("m");
        int n = str.indexOf("n");
        int o = str.indexOf("o");
        int p = str.indexOf("p");
        int q = str.indexOf("q");
        int r = str.indexOf("r");
        int s = str.indexOf("s");
        int t = str.indexOf("t");
        int u = str.indexOf("u");
        int v = str.indexOf("v");
        int w = str.indexOf("w");
        int x = str.indexOf("x");
        int y = str.indexOf("y");
        int z = str.indexOf("z");

        // Display the results of all the indexOf method calls.
        System.out.println(" a  b c  d e  f  g h i  j");
        System.out.println("=======================");
        System.out.println(a + " " + b + " " + c + " " + d + " " +
                           e + " " + f + " " + g + " " + h + " " +
                           i + " " + j + "\n");
```

```
        System.out.println("k  l  m  n  o  p q  r  s  t");
        System.out.println("==========================");
        System.out.println(k + " " + l + " " + m + " " + n + " " +
                           o + " " + p + " " + q + " " + r + " " +
                           s + " " + t + "\n");

        System.out.println("u  v  w  x  y  z");
        System.out.println("===============");
        System.out.println(u + " " + v + " " + w + " " + x + " " +
                           y + " " + z);
    }
}

/*
Output:
 a  b c  d e  f  g h i  j
========================
36 10 7 40 2 16 42 1 6 20

k  l  m  n  o  p q  r  s  t
==========================
8 35 22 14 12 23 4 11 24 31

u  v  w  x  y  z
===============
5 27 13 18 38 37
*/
```

See Also

Reference

java.lang.String

java.lang.String.intern()

Places a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 in the global string pool. This table maintains a single reference to each unique **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** ever created by an instance of the runtime in order to optimize space.

J#

```
public String intern()
```

Property Value/Return Value

An interned **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** – that is, one that has an entry in the global **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** pool. If the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** is not already in the global **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** pool, then it will be added.

Example

```
// string_intern.jsl

public class Program
{
    public static void main(String[] args)
    {
        // Create three strings in three different ways.
        String s1 = "J# Sample";
        String s2 = new StringBuffer("J#").append(" Sample").toString();
        String s3 = s2.intern();

        // Determine which strings are equivalent using the ==
        // operator (as compared to calling equals(), which is
        // a more expensive operation.
        System.out.println("s1 == s2? " + (s1 == s2));
        System.out.println("s1 == s3? " + (s1 == s3));
    }
}

/*
Output:
s1 == s2? false
s1 == s3? true
*/
```

See Also

Reference

java.lang.String

java.lang.String.lastIndexOf(int ch, int offset)

Gets the last index of a character within a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 after an offset.

J#

```
public int lastIndexOf(int ch, int offset)
```

Parameters

Parameter	Description
ch	The character to search the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for.
offset	The position within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to begin searching for the character. This value cannot be less than 0.

Property Value/Return Value

The index of the last occurrence of *ch* in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** after the offset, or -1 if the character does not exist within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** or if an invalid offset was provided.

Example

```
// string_lastindexof1.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "The quick brown fox jumps over the lazy dog.";

        // Get the index of all the characters of the alphabet
        // starting from the beginning of the String.
        int a = str.lastIndexOf('a', str.length() - 1);
        int b = str.lastIndexOf('b', str.length() - 1);
        int c = str.lastIndexOf('c', str.length() - 1);
        int d = str.lastIndexOf('d', str.length() - 1);
        int e = str.lastIndexOf('e', str.length() - 1);
        int f = str.lastIndexOf('f', str.length() - 1);
        int g = str.lastIndexOf('g', str.length() - 1);
        int h = str.lastIndexOf('h', str.length() - 1);
        int i = str.lastIndexOf('i', str.length() - 1);
        int j = str.lastIndexOf('j', str.length() - 1);
        int k = str.lastIndexOf('k', str.length() - 1);
        int l = str.lastIndexOf('l', str.length() - 1);
        int m = str.lastIndexOf('m', str.length() - 1);
        int n = str.lastIndexOf('n', str.length() - 1);
        int o = str.lastIndexOf('o', str.length() - 1);
        int p = str.lastIndexOf('p', str.length() - 1);
        int q = str.lastIndexOf('q', str.length() - 1);
        int r = str.lastIndexOf('r', str.length() - 1);
        int s = str.lastIndexOf('s', str.length() - 1);
        int t = str.lastIndexOf('t', str.length() - 1);
        int u = str.lastIndexOf('u', str.length() - 1);
        int v = str.lastIndexOf('v', str.length() - 1);
        int w = str.lastIndexOf('w', str.length() - 1);
        int x = str.lastIndexOf('x', str.length() - 1);
        int y = str.lastIndexOf('y', str.length() - 1);
        int z = str.lastIndexOf('z', str.length() - 1);

        // Display the results of all the lastIndexOf method calls.
        System.out.println(" a  b c  d  e  f  g  h i  j");
        System.out.println("==========================");
        System.out.println(a + " " + b + " " + c + " " + d + " " +
```

```
                        e + " " + f + " " + g + " " + h + " " +
                        i + " " + j + "\n");

        System.out.println("k  l  m  n  o  p q  r  s  t");
        System.out.println("==========================");
        System.out.println(k + " " + l + " " + m + " " + n + " " +
                        o + " " + p + " " + q + " " + r + " " +
                        s + " " + t + "\n");

        System.out.println(" u  v  w  x  y  z");
        System.out.println("================");
        System.out.println(u + " " + v + " " + w + " " + x + " " +
                        y + " " + z);
    }
}

/*
Output:
 a  b c  d  e  f  g  h i  j
==========================
36 10 7 40 33 16 42 32 6 20

k  l  m  n  o  p q  r  s  t
==========================
8 35 22 14 41 23 4 29 24 31

 u  v  w  x  y  z
================
21 27 13 18 38 37
*/
```

See Also

Reference

java.lang.String

java.lang.String.lastIndexOf(int ch)

Gets the last index of a character within a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4.

J#

```
public int lastIndexOf(int ch)
```

Parameters

Parameter	Description
ch	The character to search the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for.

Property Value/Return Value

The index of the last occurrence of *ch* in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**, or -1 if the character does not exist within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Example

```
// string_lastindexof2.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "The quick brown fox jumps over the lazy dog.";

        // Get the index of all the characters of the alphabet
        // starting from the beginning of the String.
        int a = str.lastIndexOf('a');
        int b = str.lastIndexOf('b');
        int c = str.lastIndexOf('c');
        int d = str.lastIndexOf('d');
        int e = str.lastIndexOf('e');
        int f = str.lastIndexOf('f');
        int g = str.lastIndexOf('g');
        int h = str.lastIndexOf('h');
        int i = str.lastIndexOf('i');
        int j = str.lastIndexOf('j');
        int k = str.lastIndexOf('k');
        int l = str.lastIndexOf('l');
        int m = str.lastIndexOf('m');
        int n = str.lastIndexOf('n');
        int o = str.lastIndexOf('o');
        int p = str.lastIndexOf('p');
        int q = str.lastIndexOf('q');
        int r = str.lastIndexOf('r');
        int s = str.lastIndexOf('s');
        int t = str.lastIndexOf('t');
        int u = str.lastIndexOf('u');
        int v = str.lastIndexOf('v');
        int w = str.lastIndexOf('w');
        int x = str.lastIndexOf('x');
        int y = str.lastIndexOf('y');
        int z = str.lastIndexOf('z');

        // Display the results of all the lastIndexOf method calls.
        System.out.println(" a  b c  d  e  f  g  h i  j");
        System.out.println("==========================");
        System.out.println(a + " " + b + " " + c + " " + d + " " +
                           e + " " + f + " " + g + " " + h + " " +
                           i + " " + j + "\n");

        System.out.println("k  l  m  n  o  p q  r  s  t");
```

```
        System.out.println("============================");
        System.out.println(k + " " + l + " " + m + " " + n + " " +
                           o + " " + p + " " + q + " " + r + " " +
                           s + " " + t + "\n");

        System.out.println(" u  v  w  x  y  z");
        System.out.println("================");
        System.out.println(u + " " + v + " " + w + " " + x + " " +
                           y + " " + z);
    }
}

/*
Output:
 a  b c  d  e  f  g  h i  j
============================
36 10 7 40 33 16 42 32 6 20

k  l  m  n  o  p q  r  s  t
============================
8 35 22 14 41 23 4 29 24 31

 u  v  w  x  y  z
================
21 27 13 18 38 37
*/
```

See Also

Reference

java.lang.String

java.lang.String.lastIndexOf(String str, int offset)

Gets the last index of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 within a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** after an offset.

J#

```
public int lastIndexOf(String str, int offset)
```

Parameters

Parameter	Description
str	The **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to search the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for.
offset	The position within the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to begin searching for the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. This value cannot be less than 0.

Property Value/Return Value

The index of the last occurrence of *str* in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** after the offset, or -1 if the character does not exist within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** or if an invalid offset was provided.

Example

```
// string_lastindexof3.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "The quick brown fox jumps over the lazy dog.";

        // Get the index of all the characters of the alphabet
        // starting from the beginning of the String.
        int a = str.lastIndexOf("a", str.length() - 1);
        int b = str.lastIndexOf("b", str.length() - 1);
        int c = str.lastIndexOf("c", str.length() - 1);
        int d = str.lastIndexOf("d", str.length() - 1);
        int e = str.lastIndexOf("e", str.length() - 1);
        int f = str.lastIndexOf("f", str.length() - 1);
        int g = str.lastIndexOf("g", str.length() - 1);
        int h = str.lastIndexOf("h", str.length() - 1);
        int i = str.lastIndexOf("i", str.length() - 1);
        int j = str.lastIndexOf("j", str.length() - 1);
        int k = str.lastIndexOf("k", str.length() - 1);
        int l = str.lastIndexOf("l", str.length() - 1);
        int m = str.lastIndexOf("m", str.length() - 1);
        int n = str.lastIndexOf("n", str.length() - 1);
        int o = str.lastIndexOf("o", str.length() - 1);
        int p = str.lastIndexOf("p", str.length() - 1);
        int q = str.lastIndexOf("q", str.length() - 1);
        int r = str.lastIndexOf("r", str.length() - 1);
        int s = str.lastIndexOf("s", str.length() - 1);
        int t = str.lastIndexOf("t", str.length() - 1);
        int u = str.lastIndexOf("u", str.length() - 1);
        int v = str.lastIndexOf("v", str.length() - 1);
        int w = str.lastIndexOf("w", str.length() - 1);
        int x = str.lastIndexOf("x", str.length() - 1);
        int y = str.lastIndexOf("y", str.length() - 1);
        int z = str.lastIndexOf("z", str.length() - 1);

        // Display the results of all the lastIndexOf method calls.
        System.out.println(" a  b c  d  e  f  g  h i  j");
        System.out.println("==========================");
```

```
        System.out.println(a + " " + b + " " + c + " " + d + " " +
                           e + " " + f + " " + g + " " + h + " " +
                           i + " " + j + "\n");

        System.out.println("k  l  m  n  o  p q  r  s  t");
        System.out.println("==========================");
        System.out.println(k + " " + l + " " + m + " " + n + " " +
                           o + " " + p + " " + q + " " + r + " " +
                           s + " " + t + "\n");

        System.out.println(" u  v  w  x  y  z");
        System.out.println("================");
        System.out.println(u + " " + v + " " + w + " " + x + " " +
                           y + " " + z);
    }
}

/*
Output:
 a  b c  d  e  f  g  h i  j
==========================
36 10 7 40 33 16 42 32 6 20

k  l  m  n  o  p q  r  s  t
==========================
8 35 22 14 41 23 4 29 24 31

 u  v  w  x  y  z
================
21 27 13 18 38 37
*/
```

See Also

Reference

java.lang.String

java.lang.String.lastIndexOf(String str)

Gets the last index of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 within a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

J#

```
public int lastIndexOf(String str)
```

Parameters

Parameter	Description
str	The **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to search the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** for.

Property Value/Return Value

The index of the last occurrence of *str* in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**, or -1 if the character does not exist within the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Example

```
// string_lastindexof4.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "The quick brown fox jumps over the lazy dog.";

        // Get the index of all the characters of the alphabet
        // starting from the beginning of the String.
        int a = str.lastIndexOf("a");
        int b = str.lastIndexOf("b");
        int c = str.lastIndexOf("c");
        int d = str.lastIndexOf("d");
        int e = str.lastIndexOf("e");
        int f = str.lastIndexOf("f");
        int g = str.lastIndexOf("g");
        int h = str.lastIndexOf("h");
        int i = str.lastIndexOf("i");
        int j = str.lastIndexOf("j");
        int k = str.lastIndexOf("k");
        int l = str.lastIndexOf("l");
        int m = str.lastIndexOf("m");
        int n = str.lastIndexOf("n");
        int o = str.lastIndexOf("o");
        int p = str.lastIndexOf("p");
        int q = str.lastIndexOf("q");
        int r = str.lastIndexOf("r");
        int s = str.lastIndexOf("s");
        int t = str.lastIndexOf("t");
        int u = str.lastIndexOf("u");
        int v = str.lastIndexOf("v");
        int w = str.lastIndexOf("w");
        int x = str.lastIndexOf("x");
        int y = str.lastIndexOf("y");
        int z = str.lastIndexOf("z");

        // Display the results of all the lastIndexOf method calls.
        System.out.println(" a  b c  d  e  f  g  h i  j");
        System.out.println("==========================");
        System.out.println(a + " " + b + " " + c + " " + d + " " +
                           e + " " + f + " " + g + " " + h + " " +
                           i + " " + j + "\n");
```

```
        System.out.println("k  l  m  n  o  p q  r  s  t");
        System.out.println("==========================");
        System.out.println(k + " " + l + " " + m + " " + n + " " +
                           o + " " + p + " " + q + " " + r + " " +
                           s + " " + t + "\n");

        System.out.println(" u  v  w  x  y  z");
        System.out.println("================");
        System.out.println(u + " " + v + " " + w + " " + x + " " +
                           y + " " + z);
    }
}

/*
Output:
 a  b c  d  e  f  g  h i  j
==========================
36 10 7 40 33 16 42 32 6 20

k  l  m  n  o  p q  r  s  t
==========================
8 35 22 14 41 23 4 29 24 31

 u  v  w  x  y  z
================
21 27 13 18 38 37
*/
```

See Also

Reference

java.lang.String

java.lang.String.length()

Gets the length of the fa4e6a2d-54fd-4a79-8103-2fae784c7bf4, which is the number of Unicode characters in the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

J#

```
public int length()
```

Property Value/Return Value

The length of the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Example

```
// string_length.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "How long is this String? ";

        // Get the length of str.
        int len = str.length();

        // Display the results of calling length.
        System.out.println(str + len);
    }
}

/*
Output:
How long is this String? 25
*/
```

See Also

Reference

java.lang.String

java.lang.String.regionMatches(int offset1, String str2, int offset2, int count)

Determines whether a region in the current fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 matches a region in another **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

J#

```
public boolean regionMatches(int offset1, String str2, int offset2, int count)
```

Parameters

Parameter	Description
offset1	The index of the first character in the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** used for comparing to *str2*.
str2	The **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to compare against the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
offset2	The index of the first character in *str2* used for comparing to the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
count	The number of characters within the two **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects to compare.

Property Value/Return Value

true if the specified regions in the two **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects match; **false** if the specified regions do not match or if *offset1*, *offset2*, or *count* are invalid.

Example

```
// string_regionmatches1.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str1 = "Missouri Florida Colorado Washington";
        String str2 = "Washington Colorado Florida Missouri";

        // Determine whether characters 0 through 7 in str1
        // match characters 28 through 35 in str2.
        boolean match1 = str1.regionMatches(0, str2, 28, 8);

        // Determine whether characters 9 through 15 in str1
        // match characters 9 through 15 in str2.
        boolean match2 = str1.regionMatches(9, str2, 9, 8);

        // Display the results of the regionMatches method calls.
        System.out.println("str1[0 - 7] == str2[28 - 35]? " + match1);
        System.out.println("str1[9 - 15] == str2[9 - 15]? " + match2);
    }
}

/*
Output:
str1[0 - 7] == str2[28 - 35]? true
str1[9 - 15] == str2[9 - 15]? false
*/
```

See Also

Reference

java.lang.String

java.lang.String.regionMatches(boolean ignoreCase, int offset1, String str2, int offset2, int count)

Determines whether a region in the current fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 matches a region in another **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**, potentially ignoring the case of the characters.

J#

```
public boolean regionMatches(boolean ignoreCase, int offset1, String str2, int offset2, int
count)
```

Parameters

Parameter	Description
ignoreCase	**true** to ignore the case of the characters when doing a comparison; **false** otherwise.
offset1	The index of the first character in the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** used for comparing to *str2*.
str2	The **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to compare against the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
offset2	The index of the first character in *str2* used for comparing to the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
count	The number of characters within the two **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects to compare.

Property Value/Return Value

true if the specified regions in the two **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects match according to the value of the *ignoreCase* flag; **false** if the specified regions do not match or if *offset1*, *offset2*, or *count* are invalid.

Example

```
// string_regionmatches2.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str1 = "Missouri Florida Colorado Washington";
        String str2 = "washington colorado florida missouri";

        // Determine whether characters 0 through 7 in str1
        // match characters 28 through 35 in str2, ignoring the
        // case of the letters.
        boolean match1 = str1.regionMatches(true, 0, str2, 28, 8);

        // Determine whether characters 9 through 15 in str1
        // match characters 9 through 15 in str2, ignoring the
        // case of the letters.
        boolean match2 = str1.regionMatches(true, 9, str2, 9, 8);

        // Display the results of the regionMatches method calls.
        System.out.println("str1[0 - 7] == str2[28 - 35]? " + match1);
        System.out.println("str1[9 - 15] == str2[9 - 15]? " + match2);
    }
}
```

```
/*
Output:
str1[0 - 7] == str2[28 - 35]? true
str1[9 - 15] == str2[9 - 15]? false
*/
```

See Also

Reference

java.lang.String

java.lang.String.replace(char oldChar, char newChar)

Replaces all existing occurrences of a character in a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 with another character.

J#

```
public String replace(char oldChar, char newChar)
```

Parameters

Parameter	Description
oldChar	The existing character to replace.
newChar	The character to replace *oldChar* with.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object where all existing occurrences of *oldChar* have been replaced with *newChar*.

Example

```
// string_replace.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "The quick brown fox jumps over the lazy dog.";

        // Replace all the 'd' characters with 'l' characters.
        String newStr = str.replace('d', 'l');

        // Display the strings for comparison.
        System.out.println("old = " + str);
        System.out.println("new = " + newStr);
    }
}

/*
Output:
old = The quick brown fox jumps over the lazy dog.
new = The quick brown fox jumps over the lazy log.
*/
```

See Also

Reference

java.lang.String

java.lang.String.startsWith(String prefix)

Determines whether the current fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 starts with the contents of another **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

J#

```
public boolean startsWith(String prefix)
```

Parameters

Parameter	Description
prefix	The **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** whose contents will be compared against the beginning of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

true if the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** starts with the contents of *prefix*; **false** otherwise.

Example

```
// string_startswith1.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str1 = "Fred is my name.";
        String str2 = "Xiao Mei is my name.";

        // The String to check the above two Strings to see
        // if they start with this value (Xi).
        String startStr = "Xi";

        // Do either of the first two Strings start with startStr?
        boolean starts1 = str1.startsWith(startStr);
        boolean starts2 = str2.startsWith(startStr);

        // Display the results of the startsWith calls.
        System.out.println("\"" + str1 + "\" starts with " +
            "\"" + startStr + "\"? " + starts1);
        System.out.println("\"" + str2 + "\" starts with " +
            "\"" + startStr + "\"? " + starts2);
    }
}

/*
Output:
"Fred is my name." starts with "Xi"? false
"Xiao Mei is my name." starts with "Xi"? true
*/
```

See Also

Reference

java.lang.String

java.lang.String.startsWith(String prefix, int offset)

Determines whether the current fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 starts with the contents of another **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

J#

```
public boolean startsWith (String prefix, int offset)
```

Parameters

Parameter	Description
prefix	The **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** whose contents will be compared against the beginning of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
offset	The index of the first character within the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to compare against *prefix*.

Property Value/Return Value

true if the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** starts with the contents of *prefix* after the offset; **false** if the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** does not start with the contents of *prefix* after the offset or if *offset* is invalid.

Example

```
// string_startswith2.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str1 = "abcFred is my name.";
        String str2 = "abcXiao Mei is my name.";

        // The String to check the above two Strings to see
        // if they start with the value Xi.
        String startStr = "Xi";

        // Do either of the first two Strings start with startStr
        // starting from index 3?
        boolean starts1 = str1.startsWith(startStr, 3);
        boolean starts2 = str2.startsWith(startStr, 3);

        // Display the results of the startsWith calls.
        System.out.println("\"" + str1 + "\" starts with " +
            "\"" + startStr + "\" starting from index 3? " + starts1);
        System.out.println("\"" + str2 + "\" starts with " +
            "\"" + startStr + "\" starting from index 3? " + starts2);
    }
}

/*
Output:
"abcFred is my name." starts with "Xi" starting from index 3? false
"abcXiao Mei is my name." starts with "Xi" starting from index 3? true
*/
```

See Also

Reference

java.lang.String

java.lang.String.substring(int offset)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the contents of the existing **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** after an offset.

J#

```
public String substring(int offset)
```

Parameters

Parameter	Description
offset	The index of the first character within the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** to copy into the new String object. A 5d87a3c3-8577-4fe2-8616-c28146468df9 is thrown if this index is invalid.

Property Value/Return Value

A new **String** object containing the contents of the existing **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** after an offset.

Example

```
// string_substring1.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "Navy blue is my favorite color.";

        // Get a substring of the above string starting from
        // index 5.
        String newStr = str.substring(5);

        // Display the two strings for comparison.
        System.out.println("old = " + str);
        System.out.println("new = " + newStr);
    }
}

/*
Output:
old = Navy blue is my favorite color.
new = blue is my favorite color.
*/
```

See Also

Reference

java.lang.String

java.lang.String.substring(int offset, int endIndex)

Creates a new **String** object containing the contents of the existing **String** between two indices.

J#

```
public String substring(int offset, int endIndex)
```

Parameters

Parameter	Description
offset	The index of the first character within the current **String** to copy into the new String object. An IndexOutofBoundsException is thrown if this index is invalid.
endIndex	The index of the last character within the current **String** to copy into the new **String** object. An IndexOutofBoundsException is thrown if this index is invalid.

Property Value/Return Value

A new **String** object containing the contents of the existing String between *offset* and *endIndex*.

Example

```
// string_substring2.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "Navy blue is my favorite color.";

        // Get a substring of the above string starting from
        // index 5 and ending at index 24.
        String newStr = str.substring(5, 24);

        // Display the two strings for comparison.
        System.out.println("old = " + str);
        System.out.println("new = " + newStr);
    }
}

/*
Output:
old = Navy blue is my favorite color.
new = blue is my favorite
*/
```

See Also

Reference

java.lang.String

java.lang.String.toCharArray()

Creates a character array containing the contents of the fa4e6a2d-54fd-4a79-8103-2fae784c7bf4.

J#

```
public char[] toCharArray()
```

Property Value/Return Value

A character array containing the contents of the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Example

```
// string_tochararray.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "This is a String.";

        // Convert the above string to a char array.
        char[] arr = str.toCharArray();

        // Display the contents of the char array.
        System.out.println(arr);
    }
}

/*
Output:
This is a String.
*/
```

See Also

Reference

java.lang.String

java.lang.String.toString()

Displays a human-readable representation of a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object.

J#

```
public String toString()
```

Property Value/Return Value

A human-readable representation of a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object. For **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects, a new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** is created containing the contents of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Example

```
// string_tostring.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "This is a String.";

        // Convert the above string to another string.
        String str2 = str.toString();

        // Display the contents of the new string.
        System.out.println(str2);
    }
}

/*
Output:
This is a String.
*/
```

See Also

Reference

java.lang.String

java.lang.String.toLowerCase()

Converts all the characters in a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 to lowercase.

J#

```
public String toLowerCase()
```

Property Value/Return Value

A **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing all the characters of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** converted to lowercase.

Example

```
// string_tolowercase1.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "ThIs Is HaRd To ReAd.";

        // Convert the above string to all lowercase.
        String lowerStr = str.toLowerCase();

        // Display the two strings for comparison.
        System.out.println("old = " + str);
        System.out.println("lowercase = " + lowerStr);
    }
}

/*
Output:
old = ThIs Is HaRd To ReAd.
lowercase = this is hard to read.
*/
```

See Also

Reference

java.lang.String

java.lang.String.toLowerCase(Locale loc)

Converts all the characters in a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 to lowercase using the given 3a8c3232-bb68-4464-b01b-9232ce6f8ca1.

J#

```
public String toLowerCase(Locale loc)
```

Parameters

Parameter	Description
loc	The **3a8c3232-bb68-4464-b01b-9232ce6f8ca1** for the newly created String object.

Property Value/Return Value

A **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing all the characters of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** converted to lowercase.

Example

```
// string_tolowercase2.jsl

import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        String str = "ThIs Is HaRd To ReAd.";

        // Convert the above string to all lowercase
        // using the default system Locale.
        Locale loc = Locale.getDefault();
        String lowerStr = str.toLowerCase(loc);

        // Display the two strings for comparison.
        System.out.println("old = " + str);
        System.out.println("lowercase = " + lowerStr);
    }
}

/*
Output:
old = ThIs Is HaRd To ReAd.
lowercase = this is hard to read.
*/
```

See Also

Reference

java.lang.String

java.lang.String.toUpperCase()

Converts all the characters in a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 to uppercase.

J#

```
public String toUpperCase()
```

Property Value/Return Value

A **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing all the characters of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** converted to uppercase.

Example

```
// string_touppercase1.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "ThIs Is HaRd To ReAd.";

        // Convert the above string to all uppercase.
        String upperStr = str.toUpperCase();

        // Display the two strings for comparison.
        System.out.println("old = " + str);
        System.out.println("uppercase = " + upperStr);
    }
}

/*
Output:
old = ThIs Is HaRd To ReAd.
uppercase = THIS IS HARD TO READ.
*/
```

See Also

Reference

java.lang.String

java.lang.String.toUpperCase(Locale loc)

Converts all the characters in a fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 to uppercase using the given 3a8c3232-bb68-4464-b01b-9232ce6f8ca1.

J#

```
public String toUpperCase(Locale loc)
```

Parameters

Parameter	Description
loc	The **3a8c3232-bb68-4464-b01b-9232ce6f8ca1** for the newly created String object.

Property Value/Return Value

A **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing all the characters of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** converted to uppercase.

Example

```
// string_touppercase2.jsl

import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        String str = "ThIs Is HaRd To ReAd.";

        // Convert the above string to all uppercase
        // using the default system Locale.
        Locale loc = Locale.getDefault();
        String upperStr = str.toUpperCase(loc);

        // Display the two strings for comparison.
        System.out.println("old = " + str);
        System.out.println("uppercase = " + upperStr);
    }
}

/*
Output:
old = ThIs Is HaRd To ReAd.
uppercase = THIS IS HARD TO READ.
*/
```

See Also

Reference

java.lang.String

java.lang.String.trim()

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 by trimming any leading or trailing whitespace from the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

J#

```
public String trim()
```

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** containing the contents of the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** minus any leading or trailing whitespace.

Example

```
// string_trim.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str = "\n\t This is a String.\n\n";

        // Trim the whitespace from the front and back of the
        // String.
        String newStr = str.trim();

        // Display the strings for comparison.
        System.out.println("old = " + str);
        System.out.println("new = " + newStr);
    }
}

/*
Output:
old =
 This is a String.


new = This is a String.
*/
```

See Also

Reference

java.lang.String

java.lang.String.valueOf(boolean bool)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the string representation of a **boolean**.

J#

```
public static String valueOf(boolean bool)
```

Parameters

Parameter	Description
bool	A **boolean** value to be converted to a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing the string representation of a **boolean**.

Example

```
// string_valueof1.jsl

public class Program
{
    public static void main(String[] args)
    {
        String trueStr = String.valueOf(true);
        String falseStr = String.valueOf(false);

        // Display the string representations of the above
        // boolean values.
        System.out.println(trueStr);
        System.out.println(falseStr);
    }
}

/*
Output:
true
false
*/
```

See Also

Reference

java.lang.String

java.lang.String.valueOf(char ch)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the string representation of a **char**.

J#

```
public static String valueOf(char ch)
```

Parameters

Parameter	Description
ch	A **char** value to be converted to a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing the string representation of a **char**.

Example

```
// string_valueof2.jsl

public class Program
{
    public static void main(String[] args)
    {
        String aStr = String.valueOf('a');
        String bStr = String.valueOf('b');

        // Display the string representations of the above
        // char values.
        System.out.println(aStr);
        System.out.println(bStr);
    }
}

/*
Output:
a
b
*/
```

See Also

Reference

java.lang.String

java.lang.String.valueOf(double dnum)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the string representation of a **double**.

C#

```
public static String valueOf(double dnum)
```

Parameters

Parameter	Description
dnum	A **double** value to be converted to a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing the string representation of a **double**.

Example

```
// string_valueof3.jsl

public class Program
{
    public static void main(String[] args)
    {
        String piStr = String.valueOf(3.14);
        String minStr = String.valueOf(Double.MIN_VALUE);

        // Display the string representations of the above
        // double values.
        System.out.println(piStr);
        System.out.println(minStr);
    }
}

/*
Output:
3.14
4.94065645841247E-324
*/
```

See Also

Reference

java.lang.String

java.lang.String.valueOf(float fnum)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the string representation of a **float**.

J#

```
public static String valueOf(float fnum)
```

Parameters

Parameter	Description
fnum	A **float** value to be converted to a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing the string representation of a **float**.

Example

```
// string_valueof4.jsl

public class Program
{
    public static void main(String[] args)
    {
        String piStr = String.valueOf(3.14f);
        String minStr = String.valueOf(Float.MIN_VALUE);

        // Display the string representations of the above
        // float values.
        System.out.println(piStr);
        System.out.println(minStr);
    }
}

/*
Output:
3.14
1.401298E-45
*/
```

See Also

Reference

java.lang.String

java.lang.String.valueOf(int inum)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the string representation of an **int**.

J#

```
public static String valueOf(int inum)
```

Parameters

Parameter	Description
inum	An **int** value to be converted to a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing the string representation of an **int**.

Example

```
// string_valueof5.jsl

public class Program
{
    public static void main(String[] args)
    {
        String hundredStr = String.valueOf(100);
        String minStr = String.valueOf(Integer.MIN_VALUE);

        // Display the string representations of the above
        // int values.
        System.out.println(hundredStr);
        System.out.println(minStr);
    }
}

/*
Output:
100
-2147483648
*/
```

See Also

Reference

java.lang.String

java.lang.String.valueOf(long lnum)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the string representation of a **long**.

J#

```
public static String valueOf(long lnum)
```

Parameters

Parameter	Description
lnum	A **long** value to be converted to a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing the string representation of a **long**.

Example

```
// string_valueof6.jsl

public class Program
{
    public static void main(String[] args)
    {
        String hundredStr = String.valueOf(100L);
        String minStr = String.valueOf(Long.MIN_VALUE);

        // Display the string representations of the above
        // long values.
        System.out.println(hundredStr);
        System.out.println(minStr);
    }
}

/*
Output:
100
-9223372036854775808
*/
```

See Also

Reference

java.lang.String

java.lang.String.valueOf(Object obj)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the string representation of an b33599ff-0daf-4116-b5d8-0649793ac6b0.

J#

```
public static String valueOf(Object obj)
```

Parameters

Parameter	Description
obj	An **b33599ff-0daf-4116-b5d8-0649793ac6b0** to be converted to a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A new String object containing the string representation of an **b33599ff-0daf-4116-b5d8-0649793ac6b0**. This is equivalent to calling the toString method on the object.

Example

```
// string_valueof7.jsl

public class Program
{
    public static void main(String[] args)
    {
        Long hundred = new Long(100);
        String hundredStr = String.valueOf(hundred);

        Long minLong = new Long(Long.MIN_VALUE);
        String minStr = String.valueOf(minLong);

        // Display the string representations of the above
        // Object values.
        System.out.println(hundredStr);
        System.out.println(minStr);
    }
}

/*
Output:
100
-9223372036854775808
*/
```

See Also

Reference

java.lang.String

java.lang.String.valueOf(char[] data)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the string representation of a character array.

J#

```
public static String valueOf(char[] data)
```

Parameters

Parameter	Description
data	A character array to be converted to a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing the string representation of a character array.

Example

```
// string_valueof8.jsl

public class Program
{
    public static void main(String[] args)
    {
        char[] arr1 = new char[] { 'a', 'b', 'c' };
        String str1 = String.valueOf(arr1);

        char[] arr2 = new char[] { '1', '2', '3' };
        String str2 = String.valueOf(arr2);

        // Display the string representations of the above
        // char array values.
        System.out.println(str1);
        System.out.println(str2);
    }
}

/*
Output:
abc
123
*/
```

See Also

Reference

java.lang.String

java.lang.String.valueOf(char[] data, int offset, int count)

Creates a new fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 object containing the string representation of a character array.

J#

```
public static String valueOf(char[] data, int offset, int count)
```

Parameters

Parameter	Description
data	A character array to be converted to a **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.
offset	The first character in the character array to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. This value must be greater than zero and less than the length of the character array.
count	The number of characters within the character array, after the offset, to be written to the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**. If this value is greater than the number of characters in the character array after the offset, then only the remaining characters are copied into the **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4**.

Property Value/Return Value

A new **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object containing the string representation of a character array.

Example

```
// string_valueof9.jsl

public class Program
{
    public static void main(String[] args)
    {
        char[] arr1 = new char[] { 'a', 'b', 'c' };
        String str1 = String.valueOf(arr1, 1, 2);

        char[] arr2 = new char[] { '1', '2', '3' };
        String str2 = String.valueOf(arr2, 2, 1);

        // Display the string representations of the above
        // char array values.
        System.out.println(str1);
        System.out.println(str2);
    }
}

/*
Output:
bc
3
*/
```

See Also

Reference

java.lang.String

java.lang.String.compareTo(Object str)

Compares two fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 objects for equality.

J#

```
public int compareTo(Object str)
```

Parameters

Parameter	Description
str	A String object to compare with the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object for equality.

Property Value/Return Value

Less than zero if the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object is less than *str*, zero if the two **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects are equal, or greater than zero if *str* is greater than the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object.

Example

```
// string_comparetoobj.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str1 = "This is string 1";
        Object str2 = new String("This is string 2");

        // Compare the two strings.
        int result = str1.compareTo(str2);

        // Display the results of the comparison.
        if (result < 0)
        {
            System.out.println("\"" + str1 + "\" +
                " is less than " +
                "\"" + str2 + "\"");
        }
        else if (result == 0)
        {
            System.out.println("\"" + str1 + "\" +
                " is equal to " +
                "\"" + str2 + "\"");
        }
        else // if (result > 0)
        {
            System.out.println("\"" + str1 + "\" +
                " is greater than " +
                "\"" + str2 + "\"");
        }
    }
}

/*
Output:
"This is string 1" is less than "This is string 2"
*/
```

See Also

Reference

java.lang.String

java.lang.String.compareToIgnoreCase(String str)

Compares two fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 objects for equality, ignoring the case of the characters.

J#

```
public int compareToIgnoreCase(String str)
```

Parameters

Parameter	Description
str	A String object to compare with the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object for equality.

Property Value/Return Value

Less than zero if the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object is less than *str*, zero if the two **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** objects are equal, or greater than zero if *str* is greater than the current **fa4e6a2d-54fd-4a79-8103-2fae784c7bf4** object. The case of the characters is ignored in all cases.

Example

```
// string_comparetoignorecase.jsl

public class Program
{
    public static void main(String[] args)
    {
        String str1 = "ThIs Is StRiNg 1";
        String str2 = "This is string 2";

        // Compare the two strings.
        int result = str1.compareToIgnoreCase(str2);

        // Display the results of the comparison.
        if (result < 0)
        {
            System.out.println("\"" + str1 + "\"" +
                " is less than " +
                "\"" + str2 + "\"");
        }
        else if (result == 0)
        {
            System.out.println("\"" + str1 + "\"" +
                " is equal to " +
                "\"" + str2 + "\"");
        }
        else // if (result > 0)
        {
            System.out.println("\"" + str1 + "\"" +
                " is greater than " +
                "\"" + str2 + "\"");
        }
    }
}

/*
Output:
"ThIs Is StRiNg 1" is less than "This is string 2"
*/
```

See Also

Reference

java.lang.String

java.lang.String.CASE_INSENSITIVE_ORDER

The 25b65df4-afc9-40a2-aace-aaf678e0769a object used when comparing fa4e6a2d-54fd-4a79-8103-2fae784c7bf4 objects for equality.

J#

```
public static final Comparator CASE_INSENSITIVE_ORDER
```

Example

```
// There is no Code Example for this constant.
```

See Also

Reference

java.lang.String

java.lang.Thread

Represents an independent stream of execution within a program.

J#

```
public class Thread implements Runnable
```

Example

```
// thread_overview.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            thr2.setDaemon(false);
            thr2.setName("Thread 2");
            thr2.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            thr3.setDaemon(true);
            thr3.setName("Thread 3");
            thr3.setPriority(Thread.NORM_PRIORITY);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Wait some time for the child threads to start.
            try
            {
                Thread.sleep(500);
            }
            catch (InterruptedException ex)
            {
                // Ignore the exception here.
            }

            // Interrupt the two sub-threads.
            thr2.interrupt();
            thr3.interrupt();

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
```

```
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        try
        {
            Thread currentThread = Thread.currentThread();
            System.out.println(currentThread.getName() +
                " [daemon = " + currentThread.isDaemon() +
                ", priority = " + currentThread.getPriority() +
                "] is executing.");

            while (true)
            {
                Thread.sleep(500);
            }
        }
        catch (InterruptedException ex)
        {
            Thread currentThread = Thread.currentThread();
            System.out.println(currentThread.getName() +
                " was interrupted: " + ex.toString());
        }
    }
}

/*
Sample Output:
Starting Thread 2...
Starting Thread 3...
Thread 2 [daemon = false, priority = 10] is executing.
Active threads = 3
Thread 3 [daemon = true, priority = 5] is executing.
Thread 3 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
Thread 2 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
*/
```

See Also
Concepts
Visual J# Class Library

java.lang.Thread.Thread(ThreadGroup group, Runnable runnable, String name)

Initializes a new instance of a named 2d622dab-0fb9-4756-8474-b94e83c9aae1 object from the given 290ee7fa-aa87-4288-8644-5b2d5e6a3817. The thread will execute the code in the run method of the given 30f26323-1728-4a53-a8f6-9560bcb68bf0 object.

J#

```
public Thread(java.lang.ThreadGroup group, java.lang.Runnable runnable, String name)
```

Parameters

Parameter	Description
group	The **290ee7fa-aa87-4288-8644-5b2d5e6a3817** containing the newly created thread.
runnable	An instance of **30f26323-1728-4a53-a8f6-9560bcb68bf0** containing the code for the thread to execute.
name	The name for the thread.

See Also

Reference

java.lang.Thread

java.lang.Thread.Thread(Runnable runnable, String name)

Initializes a new instance of a named 2d622dab-0fb9-4756-8474-b94e83c9aae1 object. The thread will execute the code in the run method of the given 30f26323-1728-4a53-a8f6-9560bcb68bf0 object.

J#

```
public Thread(java.lang.Runnable runnable, String name)
```

Parameters

Parameter	Description
runnable	An instance of **30f26323-1728-4a53-a8f6-9560bcb68bf0** containing the code for the thread to execute.
name	The name for the thread.

See Also

Reference

java.lang.Thread

java.lang.Thread.Thread(ThreadGroup group, String name)

Initializes a new instance of a named 2d622dab-0fb9-4756-8474-b94e83c9aae1 object from the given 290ee7fa-aa87-4288-8644-5b2d5e6a3817.

J#

```
public Thread(java.lang.ThreadGroup group, String name)
```

Parameters

Parameter	Description
group	The **290ee7fa-aa87-4288-8644-5b2d5e6a3817** containing the newly created thread.
name	The name for the thread.

See Also

Reference

java.lang.Thread

java.lang.Thread.Thread(ThreadGroup group, Runnable runnable)

Initializes a new instance of a 2d622dab-0fb9-4756-8474-b94e83c9aae1 object from the given 290ee7fa-aa87-4288-8644-5b2d5e6a3817. The thread will execute the code in the run method of the given 30f26323-1728-4a53-a8f6-9560bcb68bf0 object.

J#

```
public Thread(java.lang.ThreadGroup group, java.lang.Runnable runnable)
```

Parameters

Parameter	Description
group	The **290ee7fa-aa87-4288-8644-5b2d5e6a3817** containing the newly created thread.
runnable	An instance of **30f26323-1728-4a53-a8f6-9560bcb68bf0** containing the code for the thread to execute.

See Also

Reference

java.lang.Thread

java.lang.Thread.Thread(String name)

Initializes a new instance of a named 2d622dab-0fb9-4756-8474-b94e83c9aae1 object.

J#

```
public Thread(String name)
```

Parameters

Parameter	Description
name	The name for the thread.

See Also

Reference

java.lang.Thread

java.lang.Thread.Thread(Runnable runnable)

Initializes a new instance of a 2d622dab-0fb9-4756-8474-b94e83c9aae1 object. The thread will execute the code in the run method of the given 30f26323-1728-4a53-a8f6-9560bcb68bf0 object.

J#

```
public Thread(java.lang.Runnable runnable)
```

Parameters

Parameter	Description
runnable	An instance of **30f26323-1728-4a53-a8f6-9560bcb68bf0** containing the code for the thread to execute.

See Also

Reference

java.lang.Thread

java.lang.Thread.Thread()

Initializes a new instance of a 2d622dab-0fb9-4756-8474-b94e83c9aae1 object.

J#

```
public Thread()
```

See Also

Reference

java.lang.Thread

java.lang.Thread.activeCount()

Gets the number of active threads from the 290ee7fa-aa87-4288-8644-5b2d5e6a3817 containing the current thread.

J#

```
public static int activeCount()
```

Property Value/Return Value

The number of active threads from the **290ee7fa-aa87-4288-8644-5b2d5e6a3817** containing the current thread.

Example

```
// thread_activecount.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
```

```
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.checkAccess()

Determines whether a thread has permission to access a resource.

J#

```
public void checkAccess()
```

Remarks

A 7dcba491-f8ac-4c05-b593-95c5a8ecda18 is thrown if the thread does not have permission to access a resource.

Example

```
// thread_checkaccess.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Check the access for these threads.
            try
            {
                thr2.checkAccess();
                System.out.println(thr2.getName() +
                    " has access.");

                thr3.checkAccess();
                System.out.println(thr3.getName() +
                    " has access.");
            }
            catch (SecurityException ex)
            {
                System.out.println("Thread doesn't have access: " +
                    ex.toString());
            }

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Thread-0 has access.
Thread-1 has access.
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.countStackFrames()

This method is not supported by J#.

J#

```
public int countStackFrames()
```

Property Value/Return Value

This method is not supported by J#.

See Also

Reference

java.lang.Thread

java.lang.Thread.currentThread()

Gets the 2d622dab-0fb9-4756-8474-b94e83c9aae1 object that is calling the current code block.

J#

```
public static java.lang.Thread currentThread()
```

Property Value/Return Value

The **2d622dab-0fb9-4756-8474-b94e83c9aae1** object that is calling the current code block.

Example

```
// thread_currentthread.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        // Display the name of the thread executing this code.
```

```
            System.out.println(Thread.currentThread().getName() +
                " has finished executing.");
        }

        private int counter = 0;
    }

    /*
    Sample Output:
    Starting Thread-0...
    Starting Thread-1...
    Active threads = 3
    Thread-0 has finished executing.
    Thread-1 has finished executing.
    */
```

See Also

Reference

java.lang.Thread

java.lang.Thread.destroy()

This method is not supported by J#.

J#

```
public void destroy()
```

Remarks

A 15d0a457-2d84-41ff-87de-4164fcf8f559 is thrown if this method is called.

See Also

Reference

java.lang.Thread

java.lang.Thread.dumpStack()

This method is not supported by J#.

J#

```
public static void dumpStack()
```

Remarks

A MethodNotSupportedException is thrown if this method is called.

See Also

Reference

java.lang.Thread

java.lang.Thread.enumerate(Thread[] threads)

Gets all the 2d622dab-0fb9-4756-8474-b94e83c9aae1 objects that belong to the same 290ee7fa-aa87-4288-8644-5b2d5e6a3817 as the current thread.

J#

```
public static int enumerate(java.lang.Thread[] threads)
```

Parameters

Parameter	Description
threads	An array of all the **2d622dab-0fb9-4756-8474-b94e83c9aae1** objects that belong to the same **290ee7fa-aa87-4288-8644-5b2d5e6a3817** as the current thread.

Property Value/Return Value

The number of threads in the same **290ee7fa-aa87-4288-8644-5b2d5e6a3817** as the current thread.

Example

```
// thread_enumerate.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Enumerate over all the threads in the same
            // ThreadGroup as this thread.
            Thread[] threads = new Thread[Thread.activeCount()];
            int numThreads = Thread.enumerate(threads);
            for (int i = 0; i < numThreads; i++)
            {
                System.out.println("Found thread " +
                    threads[i].getName());
            }

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
```

```
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Found thread main
Found thread Thread-0
Found thread Thread-1
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.getName()

Gets the name of the current thread.

J#

```
public final String getName()
```

Property Value/Return Value

The name of the current thread.

Example

```
// thread_getname.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        // Display the name of the thread executing this code.
```

```
            System.out.println(Thread.currentThread().getName() +
                " has finished executing.");
        }

        private int counter = 0;
    }

    /*
    Sample Output:
    Starting Thread-0...
    Starting Thread-1...
    Active threads = 3
    Thread-0 has finished executing.
    Thread-1 has finished executing.
    */
```

See Also

Reference

java.lang.Thread

java.lang.Thread.getPriority()

Gets the priority of the current thread.

J#

```
public final int getPriority()
```

Property Value/Return Value

The priority of the current thread.

Remarks

The default priority for a thread is defined by 31c5b51c-29cd-4f46-a943-6275f0acf0a2. The maximum priority for a thread is defined by 956ddc7b-7eb0-4857-a442-fd0d6d2d88b4. The minimum priority for a thread is defined by f8387b47-a01d-4f34-8d67-10e44c6c24e6.

Example

```
// thread_getpriority.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            thr2.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            thr3.setPriority(Thread.NORM_PRIORITY + 1);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
```

```
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        // Display the name of the thread executing this code.
        System.out.println(Thread.currentThread().getName() +
            " [priority = " + Thread.currentThread().getPriority() +
            "] has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
Thread-0 [priority = 10] has finished executing.
Thread-1 [priority = 6] has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.getThreadGroup()

Gets the 290ee7fa-aa87-4288-8644-5b2d5e6a3817 that the current thread belongs to.

J#

```
public final java.lang.ThreadGroup getThreadGroup()
```

Property Value/Return Value

The **290ee7fa-aa87-4288-8644-5b2d5e6a3817** that the current thread belongs to.

Example

```
// thread_getthreadgroup.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            thr2.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            thr3.setPriority(Thread.NORM_PRIORITY + 1);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }
```

```
        // Display the name of the thread executing this code.
        Thread currThread = Thread.currentThread();
        System.out.println(currThread.getName() +
            " [priority = " + currThread.getPriority() +
            "] belongs to ThreadGroup " +
            currThread.getThreadGroup().getName());

        System.out.println(currThread.getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
Thread-0 [priority = 10] belongs to ThreadGroup main
Thread-0 has finished executing.
Thread-1 [priority = 6] belongs to ThreadGroup main
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.interrupt()

Interrupts a thread that is in the waiting state.

J#

```
public void interrupt()
```

Remarks

An 5a482700-e124-47f0-b132-a0e034ef746f is thrown when the thread is interrupted. This exception will only be thrown if the thread is ever in a waiting state after the interrupt method is called.

Note

For Visual J# 2005, the permission sets for this method have changed. Whereas no permission was needed in earlier version s, now permission is demanded when calling this method. See SecurityAttribute and SecurityAction for more information.

Example

```
// thread_interrupt.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            thr2.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            thr3.setPriority(Thread.NORM_PRIORITY + 1);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Wait for some time for the child threads to start.
            try
            {
                Thread.sleep(500);
            }
            catch (InterruptedException ex)
            {
                // Ignore the exception here.
            }

            // Interrupt the two sub-threads.
            thr2.interrupt();
```

```
            thr3.interrupt();

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        try
        {
            Thread currentThread = Thread.currentThread();
            System.out.println(currentThread.getName() +
                " [daemon = " + currentThread.isDaemon() +
                ", priority = " + currentThread.getPriority() +
                "] is executing.");

            while (true)
            {
                Thread.sleep(500);
            }
        }
        catch (InterruptedException ex)
        {
            Thread currThread = Thread.currentThread();
            System.out.println(currThread.getName() +
                " was interrupted: " + ex.toString());
        }
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Thread-0 [daemon = false, priority = 10] is executing.
Active threads = 3
Thread-1 [daemon = false, priority = 6] is executing.
Thread-1 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
Thread-0 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.interrupted()

Determines if the current thread has been interrupted.

J#

```
public static boolean interrupted()
```

Property Value/Return Value

true if the current thread has been interrupted; **false** otherwise.

Example

```
// thread_interrupted.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            thr2.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            thr3.setPriority(Thread.NORM_PRIORITY + 1);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Wait for some time for the child threads to start.
            try
            {
                Thread.sleep(500);
            }
            catch (InterruptedException ex)
            {
                // Ignore the exception here.
            }

            // Interrupt the two sub-threads.
            if (!thr2.interrupted())
            {
                thr2.interrupt();
            }

            if (!thr3.interrupted())
            {
```

```
                thr3.interrupt();
            }

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        try
        {
            Thread currentThread = Thread.currentThread();
            System.out.println(currentThread.getName() +
                " [daemon = " + currentThread.isDaemon() +
                ", priority = " + currentThread.getPriority() +
                "] is executing.");

            while (true)
            {
                Thread.sleep(500);
            }
        }
        catch (InterruptedException ex)
        {
            Thread currThread = Thread.currentThread();
            System.out.println(currThread.getName() +
                " was interrupted: " + ex.toString());
        }
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Thread-0 [daemon = false, priority = 10] is executing.
Starting Thread-1...
Active threads = 3
Thread-1 [daemon = false, priority = 6] is executing.
Thread-0 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
Thread-1 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.isAlive()

Determines if the current thread is alive.

J#

```
public final boolean isAlive()
```

Property Value/Return Value

true if the current thread is alive; **false** otherwise.

Example

```
// thread_isalive.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Determine which threads are still alive.
            if (thr2.isAlive())
            {
                System.out.println(thr2.getName() +
                    " is alive.");
            }
            else
            {
                System.out.println(thr2.getName() +
                    " is NOT alive.");
            }

            if (thr3.isAlive())
            {
                System.out.println(thr3.getName() +
                    " is alive.");
            }
            else
            {
                System.out.println(thr3.getName() +
```

```
                    " is NOT alive.");
            }


            // Block until the other threads finish.
            thr2.join();
            thr3.join();

            // Determine again which threads are still alive.
            if (thr2.isAlive())
            {
                System.out.println(thr2.getName() +
                    " is alive.");
            }
            else
            {
                System.out.println(thr2.getName() +
                    " is NOT alive.");
            }

            if (thr3.isAlive())
            {
                System.out.println(thr3.getName() +
                    " is alive.");
            }
            else
            {
                System.out.println(thr3.getName() +
                    " is NOT alive.");
            }
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
Thread-0 is alive.
Thread-1 is alive.
Thread-0 has finished executing.
Thread-1 has finished executing.
Thread-0 is NOT alive.
Thread-1 is NOT alive.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.isDaemon()

Determines if the current thread is a daemon (background) thread.

J#

```
public final boolean isDaemon()
```

Property Value/Return Value

true if the current thread is a daemon thread; **false** otherwise.

Example

```
// thread_isdaemon.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread as a daemon thread
            // and start it.
            Thread thr3 = new Thread(this);
            thr3.setDaemon(true);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        Thread currThread = Thread.currentThread();
        System.out.println(currThread.getName() +
            " is a daemon thread? " + currThread.isDaemon());

        for (int i = 0; i < 100000000; i++)
```

```
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(currThread.getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Thread-0 is a daemon thread? false
Active threads = 3
Thread-1 is a daemon thread? true
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.isInterrupted()

Determines if the current thread has been interrupted.

J#

```
public boolean isInterrupted()
```

Property Value/Return Value

true if the current thread has been interrupted; **false** otherwise.

Example

```
// thread_isinterrupted.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            thr2.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            thr3.setPriority(Thread.NORM_PRIORITY + 1);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Wait for some time for the child threads to start.
            try
            {
                Thread.sleep(500);
            }
            catch (InterruptedException ex)
            {
                // Ignore the exception here.
            }

            // Interrupt the two sub-threads.
            if (!thr2.isInterrupted())
            {
                thr2.interrupt();
            }

            if (!thr3.isInterrupted())
            {
```

```
                thr3.interrupt();
            }

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        try
        {
            Thread currentThread = Thread.currentThread();
            System.out.println(currentThread.getName() +
                " [daemon = " + currentThread.isDaemon() +
                ", priority = " + currentThread.getPriority() +
                "] is executing.");

            while (true)
            {
                Thread.sleep(500);
            }
        }
        catch (InterruptedException ex)
        {
            Thread currThread = Thread.currentThread();
            System.out.println(currThread.getName() +
                " was interrupted: " + ex.toString());
        }
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Thread-0 [daemon = false, priority = 10] is executing.
Active threads = 3
Thread-1 [daemon = false, priority = 6] is executing.
Thread-1 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
Thread-0 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.join(long millis, int nanos)

Blocks the calling thread until a secondary thread completes execution or until the specified time has elapsed.

J#

```
public final void join(long millis, int nanos)
```

Parameters

Parameter	Description
millis	The number of milliseconds to wait until forcing a join.
nanos	The additional number of nanoseconds to wait until forcing a join.

Example

```
// thread_join1.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish or for
            // 100 ms, whichever comes first.
            thr2.join(100, 0);
            System.out.println("No longer blocking on thread " +
                thr2.getName());

            thr3.join(100, 0);
            System.out.println("No longer blocking on thread " +
                thr3.getName());
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }
```

```
    // Implements Runnable.run()
    public void run()
    {
        Thread currThread = Thread.currentThread();
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(currThread.getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
Thread-0 has finished executing.
No longer blocking on thread Thread-0
Thread-1 has finished executing.
No longer blocking on thread Thread-1
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.join(long millis)

Blocks the calling thread until a secondary thread completes execution or until the specified time has elapsed.

J#

```
public final void join(long millis)
```

Parameters

Parameter	Description
millis	The number of milliseconds to wait until forcing a join.

Example

```
// thread_join2.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish or for
            // 100 ms, whichever comes first.
            thr2.join(100);
            System.out.println("No longer blocking on thread " +
                thr2.getName());

            thr3.join(100);
            System.out.println("No longer blocking on thread " +
                thr3.getName());
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
```

```
        Thread currThread = Thread.currentThread();
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(currThread.getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
No longer blocking on thread Thread-0
Thread-0 has finished executing.
Thread-1 has finished executing.
No longer blocking on thread Thread-1
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.join()

Blocks the calling thread until a secondary thread completes execution.

J#

```
public final void join()
```

Example

```
// thread_join3.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            System.out.println("No longer blocking on thread " +
                thr2.getName());

            thr3.join();
            System.out.println("No longer blocking on thread " +
                thr3.getName());
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        Thread currThread = Thread.currentThread();
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }
```

```
            System.out.println(currThread.getName() +
                " has finished executing.");
        }

        private int counter = 0;
    }

    /*
    Sample Output:
    Starting Thread-0...
    Starting Thread-1...
    Active threads = 3
    Thread-0 has finished executing.
    Thread-1 has finished executing.
    No longer blocking on thread Thread-0
    No longer blocking on thread Thread-1
    */
```

See Also

Reference

java.lang.Thread

java.lang.Thread.resume()

Resumes execution of a thread that has been suspended.

J#

```
public final void resume()
```

Remarks

Note For Visual J# 2005, the permission sets for this method have changed. Whereas no permission was needed in earlier versions, now permission is demanded when calling this method. See SecurityAttribute and SecurityAction for more information.

Example

```
// thread_resume.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Suspend the secondary thread.
            System.out.println("Suspending " +
                thr2.getName() + "...");
            thr2.suspend();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish or until
            // the secondary thread executes for 1000 ms.
            thr2.join(1000);
            thr3.join();

            // Resume the secondary thread.
            System.out.println("Resuming " + thr2.getName());
            thr2.resume();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Suspending Thread-0...
Starting Thread-1...
Active threads = 3
Thread-1 has finished executing.
Resuming Thread-0
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.run()

Runs a thread in its entirety. The calling thread is blocked until the called thread completes execution.

J#

```
public void run()
```

Example

```
// thread_run.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        // Create a second thread and run it.
        Thread thr2 = new Thread(this);
        System.out.println("Running " +
                thr2.getName() + "...");
        thr2.run();

        // Create a third thread and run it.
        Thread thr3 = new Thread(this);
        System.out.println("Running " +
                thr3.getName() + "...");
        thr3.run();

        // Display the number of active threads.  This
        // count includes the main thread.
        System.out.println("Active threads after run = " +
            Thread.activeCount());
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Running Thread-0...
main has finished executing.
Running Thread-1...
main has finished executing.
Active threads after run = 1
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.setDaemon(boolean daemonFlag)

Sets a value indicating that the current thread is a daemon (background) thread.

J#

```
public final void setDaemon(boolean daemonFlag)
```

Parameters

Parameter	Description
daemonFlag	**true** to make the current thread a daemon thread; **false** otherwise.

Example

```
// thread_setdaemon.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread as a daemon thread
            // and start it.
            Thread thr3 = new Thread(this);
            thr3.setDaemon(true);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        Thread currThread = Thread.currentThread();
        System.out.println(currThread.getName() +
            " is a daemon thread? " + currThread.isDaemon());
```

```
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(currThread.getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Thread-0 is a daemon thread? false
Active threads = 3
Thread-1 is a daemon thread? true
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.setName(String name)

Sets the name of the current thread.

J#

```
public final void setName(String name)
```

Parameters

Parameter	Description
name	The name for the current thread.

Example

```
// thread_setname.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            thr2.setName("Thread 2");
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            thr3.setName("Thread 3");
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
```

```
        }

        // Display the name of the thread executing this code.
        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread 2...
Starting Thread 3...
Active threads = 3
Thread 2 has finished executing.
Thread 3 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.setPriority(int priority)

Sets the priority of the current thread.

J#

```
public final void setPriority(int priority)
```

Parameters

Parameter	Description
priority	A value indicating the priority for the current thread.

Remarks

The default priority for a thread is defined by 31c5b51c-29cd-4f46-a943-6275f0acf0a2. The maximum priority for a thread is defined by 956ddc7b-7eb0-4857-a442-fd0d6d2d88b4. The minimum priority for a thread is defined by f8387b47-a01d-4f34-8d67-10e44c6c24e6.

Example

```
// thread_setpriority.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            thr2.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            thr3.setPriority(Thread.NORM_PRIORITY + 1);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
```

```
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        // Display the name of the thread executing this code.
        System.out.println(Thread.currentThread().getName() +
            " [priority = " + Thread.currentThread().getPriority() +
            "] has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
Thread-0 [priority = 10] has finished executing.
Thread-1 [priority = 6] has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.sleep(long millis, int nanos)

Pauses execution of the current thread for the specified time.

J#

```
public static void sleep(long millis, int nanos)
```

Parameters

Parameter	Description
millis	The number of milliseconds to sleep.
nanos	The additional number of nanoseconds to sleep.

Example

```
// thread_sleep1.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Sleep for 5 seconds (5000 milliseconds
            // and 0 nanosecondss).
            Thread.sleep(5000, 0);

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
```

```
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Thread-0 has finished executing.
Starting Thread-1...
Active threads = 2
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.sleep(long millis)

Pauses execution of the current thread for the specified time.

J#

```
public static void sleep(long millis)
```

Parameters

Parameter	Description
millis	The number of milliseconds to sleep.

Example

```
// thread_sleep2.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Sleep for 5 seconds (5000 milliseconds).
            Thread.sleep(5000);

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
```

```
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Thread-0 has finished executing.
Starting Thread-1...
Active threads = 2
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.start()

Starts execution of a thread in the background. Control is immediately returned to the calling thread.

J#

```
public void start()
```

Example

```
// thread_start.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
```

```
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.stop(java.lang.Throwable e)

Stops execution of a thread. The given exception is passed to the thread before it is terminated.

J#

```
public final void stop(java.lang.Throwable e)
```

Parameters

Parameter	Description
e	The exception to send to the thread before it is terminated.

Remarks

Note For Visual J# 2005, the permission sets for this method have changed. Whereas no permission was needed in earlier versions, now permission is demanded when calling this method. See SecurityAttribute and SecurityAction for more information.

Example

```
// thread_stop1.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        // Create a second thread and start it.
        Thread thr2 = new Thread(this);
        System.out.println("Starting " +
                thr2.getName() + "...");
        thr2.start();

        // Create a third thread and start it.
        Thread thr3 = new Thread(this);
        System.out.println("Starting " +
                thr3.getName() + "...");
        thr3.start();

        // Display the number of active threads.  This
        // count includes the main thread.
        System.out.println("Active threads = " +
            Thread.activeCount());

        // The threads are taking a long time.
        thr2.stop(new InterruptedException("Speed it up!"));
        System.out.println(thr2.getName() +
            " has been manually stopped.");

        thr3.stop(new InterruptedException("Speed it up!"));
        System.out.println(thr3.getName() +
            " has been manually stopped.");
    }

    // Implements Runnable.run()
    public void run()
    {
        try
        {
```

```
            for (int i = 0; i < Integer.MAX_VALUE; i++)
            {
                // Just increment a counter.
                counter++;
            }

            System.out.println(Thread.currentThread().getName() +
                " has finished executing.");
        }
        catch (Exception ex)
        {
            System.out.println(Thread.currentThread().getName() +
                " received an exception: " +
                ex.toString());
        }
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
Thread-0 has been manually stopped.
Thread-0 received an exception: java.lang.InterruptedException: Speed it up!
Thread-1 received an exception: java.lang.InterruptedException: Speed it up!
Thread-1 has been manually stopped.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.stop()

Stops execution of a thread.

J#

```
public final void stop()
```

Remarks

Note For Visual J# 2005, the permission sets for this method have changed. Whereas no permission was needed in earlier versions, now permission is demanded when calling this method. See SecurityAttribute and SecurityAction for more information.

Example

```
// thread_stop2.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        // Create a second thread and start it.
        Thread thr2 = new Thread(this);
        System.out.println("Starting " +
                thr2.getName() + "...");
        thr2.start();

        // Create a third thread and start it.
        Thread thr3 = new Thread(this);
        System.out.println("Starting " +
                thr3.getName() + "...");
        thr3.start();

        // Display the number of active threads.  This
        // count includes the main thread.
        System.out.println("Active threads = " +
            Thread.activeCount());

        // The threads are taking a long time.
        thr2.stop();
        System.out.println(thr2.getName() +
            " has been manually stopped.");

        thr3.stop();
        System.out.println(thr3.getName() +
            " has been manually stopped.");
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < Integer.MAX_VALUE; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
```

```
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads = 3
Thread-0 has been manually stopped.
Thread-1 has been manually stopped.
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.suspend()

Suspends execution of a thread until resume is called.

J#

```
public final void suspend()
```

Remarks

Note For Visual J# 2005, the permission sets for this method have changed. Whereas no permission was needed in earlier versions, now permission is demanded when calling this method. See SecurityAttribute and SecurityAction for more information.

Example

```
// thread_suspend.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Suspend the secondary thread.
            System.out.println("Suspending " +
                thr2.getName() + "...");
            thr2.suspend();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish or until
            // the secondary thread executes for 5000 ms.
            thr2.join(5000);
            thr3.join();

            // Resume the secondary thread.
            System.out.println("Resuming " + thr2.getName());
            thr2.resume();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
```

```
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Suspending Thread-0...
Starting Thread-1...
Active threads = 3
Thread-1 has finished executing.
Resuming Thread-0
*/
```

See Also

Reference

java.lang.Thread

java.lang.Thread.yield()

Yields execution control to another thread.

J#

```
public static void yield()
```

Example

```
// thread_yield.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a second thread and start it.
            Thread thr2 = new Thread(this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads.  This
            // count includes the main thread.
            System.out.println("Active threads = " +
                Thread.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 10000000; i++)
        {
            // Yield control to another thread every
            // 1000000 iterations.
            if ((i % 1000000) == 0)
            {
                System.out.println(Thread.currentThread().getName() +
                    " is yielding control...");
                Thread.currentThread().yield();
            }
        }
```

```
            System.out.println(Thread.currentThread().getName() +
                " has finished executing.");
        }

        private int counter = 0;
    }

    /*
    Sample Output:
    Starting Thread-0...
    Starting Thread-1...
    Thread-0 is yielding control...
    Active threads = 3
    Thread-1 is yielding control...
    Thread-0 is yielding control...
    Thread-1 is yielding control...
    Thread-0 is yielding control...
    Thread-1 is yielding control...
    Thread-0 is yielding control...
    Thread-1 is yielding control...
    Thread-0 is yielding control...
    Thread-1 is yielding control...
    Thread-0 is yielding control...
    Thread-1 is yielding control...
    Thread-1 is yielding control...
    Thread-0 is yielding control...
    Thread-1 is yielding control...
    Thread-0 is yielding control...
    Thread-1 is yielding control...
    Thread-0 is yielding control...
    Thread-1 is yielding control...
    Thread-0 is yielding control...
    Thread-1 has finished executing.
    Thread-0 has finished executing.
    */
```

See Also

Reference

java.lang.Thread

java.lang.Thread.MAX_PRIORITY

A constant representing the maximum priority that can be assigned to a thread.

J#

```
public static final int MAX_PRIORITY
```

See Also

Reference

java.lang.Thread

java.lang.Thread.MIN_PRIORITY

A constant representing the minimum priority that can be assigned to a thread.

J#

```
public static final int MIN_PRIORITY
```

See Also

Reference

java.lang.Thread

java.lang.Thread.NORM_PRIORITY

A constant representing the normal priority that is assigned to newly created threads if a priority is not manually set.

J#

```
public static final int NORM_PRIORITY
```

See Also

Reference

java.lang.Thread

java.lang.ThreadGroup

Represents a group of threads that share common traits.

J#

```
public class ThreadGroup
```

Example

```
// threadgroup_overview.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            parentGroup.setDaemon(false);
            parentGroup.setMaxPriority(Thread.MAX_PRIORITY - 2);

            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");
            childGroup.setDaemon(true);
            childGroup.setMaxPriority(Thread.NORM_PRIORITY);

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Enumerate over all the child groups of parentGroup.
            ThreadGroup[] groups =
                new ThreadGroup[parentGroup.activeGroupCount()];
            int numGroups = parentGroup.enumerate(groups, true);
            for (int i = 0; i < numGroups; i++)
            {
                System.out.println("Found ThreadGroup " +
                    groups[i].getName() +
                    " [Max Priority = " +
                    groups[i].getMaxPriority() +
                    ", Parent = " +
                    groups[i].getParent().getName() +
                    ", Is Daemon = " +
                    groups[i].isDaemon() + "]");
            }

            // Block until the other threads finish.
```

```
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Found ThreadGroup Child ThreadGroup
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also
Concepts
Visual J# Class Library

java.lang.ThreadGroup.ThreadGroup(ThreadGroup parent, String name)

Initializes a new instance of a named 290ee7fa-aa87-4288-8644-5b2d5e6a3817 object with the given parent.

J#

```
public ThreadGroup(java.lang.ThreadGroup parent, String name)
```

Parameters

Parameter	Description
parent	The parent ThreadGroup object for this **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.
name	The name for the **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.ThreadGroup(String name)

Initializes a new instance of a named 290ee7fa-aa87-4288-8644-5b2d5e6a3817 object.

J#

```
public ThreadGroup(String name)
```

Parameters

Parameter	Description
name	The name for the ThreadGroup.

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.ThreadGroup()

Initializes a new instance of a 290ee7fa-aa87-4288-8644-5b2d5e6a3817 object.

J#

```
public ThreadGroup()
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.activeCount()

Gets the number of active threads in this 290ee7fa-aa87-4288-8644-5b2d5e6a3817 and all child **290ee7fa-aa87-4288-8644-5b2d5e6a3817** objects.

J#

```
public int activeCount()
```

Property Value/Return Value

The number of active threads in this **290ee7fa-aa87-4288-8644-5b2d5e6a3817** and all child **290ee7fa-aa87-4288-8644-5b2d5e6a3817** objects.

Example

```
// threadgroup_activecount.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }
```

```
    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.activeGroupCount()

Gets the number of active groups that are children of this 290ee7fa-aa87-4288-8644-5b2d5e6a3817.

J#

```
public int activeGroupCount()
```

Property Value/Return Value

The number of active groups that are children of this **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Example

```
// threadgroup_activegroupcount.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active group threads (child " +
                "groups) of group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeGroupCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
```

```
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active group threads (child groups) of group "Parent ThreadGroup" = 1
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.allowThreadSuspension(boolean allow)

This method is not supported by J#.

J#

```
public boolean allowThreadSuspension(boolean allow)
```

Property Value/Return Value

This method is not supported by J#.

Note

: A MethodNotSupportedException is thrown if this method is called.

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.checkAccess()

Determines whether a 290ee7fa-aa87-4288-8644-5b2d5e6a3817 has permission to access a resource.

J#

```
public final void checkAccess()
```

Remarks

A 7dcba491-f8ac-4c05-b593-95c5a8ecda18 is thrown if the **290ee7fa-aa87-4288-8644-5b2d5e6a3817** does not have permission to access a resource.

Example

```
// threadgroup_checkaccess.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Check the access for these ThreadGroups.
            try
            {
                parentGroup.checkAccess();
                System.out.println(parentGroup.getName() +
                    " has access.");
                childGroup.checkAccess();
                System.out.println(childGroup.getName() +
                    " has access.");
            }
            catch (SecurityException ex)
            {
                System.out.println("Thread doesn't have access: " +
                    ex.toString());
            }

            // Block until the other threads finish.
```

```
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Parent ThreadGroup has access.
Child ThreadGroup has access.
Thread-1 has finished executing.
Thread-0 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.destroy()

Destroys this 290ee7fa-aa87-4288-8644-5b2d5e6a3817.

J#

```
public final void destroy()
```

Remarks

An fab94e73-821e-4c9e-8a69-b54cdecd1a3e will be thrown if there are no active threads in the **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Example

```
// threadgroup_destroy.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Block until the other threads finish.
            thr2.join();
            thr3.join();

            // Remove all threads from the ThreadGroup and
            // destroy the ThreadGroups.
            try
            {
                childGroup.destroy();
                System.out.println(childGroup.getName() +
                    " has been destroyed.");
            }
            catch (IllegalThreadStateException ex)
            {
                System.out.println(childGroup.getName() +
                    " is already empty: " +
                    ex.toString());
```

```
            }

            try
            {
                parentGroup.destroy();
                System.out.println(parentGroup.getName() +
                    " has been destroyed.");
            }
            catch (IllegalThreadStateException ex)
            {
                System.out.println(parentGroup.getName() +
                    " is already empty: " +
                    ex.toString());
            }
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Thread-0 has finished executing.
Thread-1 has finished executing.
Child ThreadGroup is already empty: java.lang.IllegalThreadStateException: The thread has b
een destroyed or the group is empty
Parent ThreadGroup is already empty: java.lang.IllegalThreadStateException: The thread has
been destroyed or the group is empty
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.enumerate(ThreadGroup[] groups, boolean recurse)

Gets all the child 290ee7fa-aa87-4288-8644-5b2d5e6a3817 objects of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

J#

```
public int enumerate(java.lang.ThreadGroup[] groups, boolean recurse)
```

Parameters

Parameter	Description
groups	An array of all the child **290ee7fa-aa87-4288-8644-5b2d5e6a3817** objects of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.
recurse	**true** to include the grandchild **290ee7fa-aa87-4288-8644-5b2d5e6a3817** objects; **false** to include only the immediate child **290ee7fa-aa87-4288-8644-5b2d5e6a3817** objects.

Property Value/Return Value

The number of child **290ee7fa-aa87-4288-8644-5b2d5e6a3817** objects of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Example

```
// threadgroup_enumerate1.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Enumerate over all the child groups of parentGroup.
            ThreadGroup[] groups =
                new ThreadGroup[parentGroup.activeGroupCount()];
```

```
            int numGroups = parentGroup.enumerate(groups, true);
            for (int i = 0; i < numGroups; i++)
            {
                System.out.println("Found ThreadGroup " +
                    groups[i].getName());
            }

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Found ThreadGroup Child ThreadGroup
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.enumerate(ThreadGroup[] groups)

Gets all the child 290ee7fa-aa87-4288-8644-5b2d5e6a3817 objects of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

J#

```
public int enumerate(java.lang.ThreadGroup[] groups)
```

Parameters

Parameter	Description
groups	An array of all the child **290ee7fa-aa87-4288-8644-5b2d5e6a3817** objects of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Property Value/Return Value

The number of child **290ee7fa-aa87-4288-8644-5b2d5e6a3817** objects of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Example

```
// threadgroup_enumerate2.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Enumerate over all the child groups of parentGroup.
            ThreadGroup[] groups =
                new ThreadGroup[parentGroup.activeGroupCount()];
            int numGroups = parentGroup.enumerate(groups);
            for (int i = 0; i < numGroups; i++)
            {
                System.out.println("Found ThreadGroup " +
                    groups[i].getName());
            }
```

```
            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Found ThreadGroup Child ThreadGroup
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.enumerate(Thread[] threads, boolean recurse)

Gets all the 2d622dab-0fb9-4756-8474-b94e83c9aae1 objects that belong to the current 290ee7fa-aa87-4288-8644-5b2d5e6a3817 and optionally the children of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

J#

```
public int enumerate(java.lang.Thread[] threads, boolean recurse)
```

Parameters

Parameter	Description
threads	An array of all the **2d622dab-0fb9-4756-8474-b94e83c9aae1** objects that belong to the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817** and optionally the children of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.
recurse	**true** to include the threads belonging to the children of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**; **false** otherwise.

Property Value/Return Value

The number of threads in the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817** and optionally the children of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Example

```
// threadgroup_enumerate3.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Enumerate over all the child threads of parentGroup.
```

```
            Thread[] threads = new Thread[parentGroup.activeCount()];
            int numThreads = parentGroup.enumerate(threads, true);
            for (int i = 0; i < numThreads; i++)
            {
                System.out.println("Found Thread " +
                    threads[i].getName());
            }

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Found Thread Thread-0
Thread-0 has finished executing.
Found Thread Thread-1
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.enumerate(Thread[] threads)

Gets all the 2d622dab-0fb9-4756-8474-b94e83c9aae1 objects that belong to the current 290ee7fa-aa87-4288-8644-5b2d5e6a3817 and the children of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

J#

```
public int enumerate(java.lang.Thread[] threads)
```

Parameters

Parameter	Description
threads	An array of all the **2d622dab-0fb9-4756-8474-b94e83c9aae1** objects that belong to the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817** and the children of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Property Value/Return Value

The number of threads in the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817** and the children of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Example

```
// threadgroup_enumerate4.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Enumerate over all the child threads of parentGroup.
            Thread[] threads = new Thread[parentGroup.activeCount()];
            int numThreads = parentGroup.enumerate(threads);
            for (int i = 0; i < numThreads; i++)
            {
                System.out.println("Found Thread " +
                    threads[i].getName());
            }
```

```
            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Found Thread Thread-0
Thread-0 has finished executing.
Found Thread Thread-1
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.getMaxPriority()

Gets the maximum priority that can be set for any thread in the 290ee7fa-aa87-4288-8644-5b2d5e6a3817.

J#

```
public final int getMaxPriority()
```

Property Value/Return Value

A value indicating the maximum priority that can be set for any thread in the **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Remarks

If an attempt is made to add a thread with a higher priority than the value set in setMaxPriority to a **290ee7fa-aa87-4288-8644-5b2d5e6a3817**, then the priority for that thread will be lowered to the value set in setMaxPriority.

The default priority for a thread is defined by 31c5b51c-29cd-4f46-a943-6275f0acf0a2. The maximum priority for a thread is defined by 956ddc7b-7eb0-4857-a442-fd0d6d2d88b4. The minimum priority for a thread is defined by f8387b47-a01d-4f34-8d67-10e44c6c24e6.

Example

```
// threadgroup_getmaxpriority.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.  Set the maximum priority for
            // the parent to 8 and for the child to 5.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            parentGroup.setMaxPriority(Thread.MAX_PRIORITY - 2);

            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");
            childGroup.setMaxPriority(Thread.NORM_PRIORITY);

            // Create a second thread with maximum priority (pri 10)
            // and start it.
            Thread thr2 = new Thread(parentGroup, this);
            thr2.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread with maximum priority (pri 10)
            // and start it.
            Thread thr3 = new Thread(childGroup, this);
            thr3.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
```

```
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        // Display what the priority was actually set to when the
        // thread was added to the ThreadGroup.
        System.out.println(Thread.currentThread().getName() +
            " [priority = " + Thread.currentThread().getPriority() +
            "] finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-0 [priority = 8] finished executing.
Thread-1 [priority = 5] finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.getName()

Gets the name for a 290ee7fa-aa87-4288-8644-5b2d5e6a3817.

J#

```
public final String getName()
```

Property Value/Return Value

The name of the **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Example

```
// threadgroup_getname.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
```

```
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-0 has finished executing.
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.getParent()

Gets the parent 290ee7fa-aa87-4288-8644-5b2d5e6a3817 of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

J#

```
public final java.lang.ThreadGroup getParent()
```

Property Value/Return Value

The parent **290ee7fa-aa87-4288-8644-5b2d5e6a3817** of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Example

```
// threadgroup_getparent.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Display the parent ThreadGroup of parentGroup
            // and childGroup.
            System.out.println("The parent ThreadGroup for " +
                parentGroup.getName() + " is " +
                parentGroup.getParent().getName());
            System.out.println("The parent ThreadGroup for " +
                childGroup.getName() + " is " +
                childGroup.getParent().getName());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
```

```
            }
            catch (InterruptedException ex)
            {
                System.out.println(ex.toString());
            }
        }

        // Implements Runnable.run()
        public void run()
        {
            for (int i = 0; i < 100000000; i++)
            {
                // Just increment a counter.
                counter++;
            }

            System.out.println(Thread.currentThread().getName() +
                " has finished executing.");
        }

        private int counter = 0;
    }

    /*
    Sample Output:
    Starting Thread-0...
    Starting Thread-1...
    Active threads in group "Parent ThreadGroup" = 2
    Thread-0 has finished executing.
    The parent ThreadGroup for Parent ThreadGroup is main
    The parent ThreadGroup for Child ThreadGroup is Parent ThreadGroup
    Thread-1 has finished executing.
    */
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.isDaemon()

Determines whether the current 290ee7fa-aa87-4288-8644-5b2d5e6a3817 is a daemon (background) **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

J#

```
public final boolean isDaemon()
```

Property Value/Return Value

true if the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817** is a daemon (background) **290ee7fa-aa87-4288-8644-5b2d5e6a3817**; **false** otherwise.

Example

```
// threadgroup_isdaemon.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            parentGroup.setDaemon(false);

            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");
            childGroup.setDaemon(true);

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Display which ThreadGroups are daemon ThreadGroups.
            System.out.println(parentGroup.getName() +
                " is a daemon ThreadGroup? " +
                parentGroup.isDaemon());
            System.out.println(childGroup.getName() +
                " is a daemon ThreadGroup? " +
                childGroup.isDaemon());
```

```
            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-0 has finished executing.
Parent ThreadGroup is a daemon ThreadGroup? false
Child ThreadGroup is a daemon ThreadGroup? true
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.isDestroyed()

Determines whether the current 290ee7fa-aa87-4288-8644-5b2d5e6a3817 has already been destroyed.

J#

```
public boolean isDestroyed()
```

Property Value/Return Value

true if the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817** has already been destroyed; **false** otherwise.

Example

```
// threadgroup_isdestroyed.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Block until the other threads finish.
            thr2.join();
            thr3.join();

            // Remove all threads from the ThreadGroup and
            // destroy the ThreadGroups.
            try
            {
                if (!childGroup.isDestroyed())
                {
                    childGroup.destroy();
                }
                else
                {
                    System.out.println(childGroup.getName() +
                        " has been destroyed.");
                }
            }
```

```
            catch (IllegalThreadStateException ex)
            {
                System.out.println(childGroup.getName() +
                    " is already empty: " +
                    ex.toString());
            }

            try
            {
                if (!parentGroup.isDestroyed())
                {
                    parentGroup.destroy();
                }
                else
                {
                    System.out.println(parentGroup.getName() +
                        " has been destroyed.");
                }
            }
            catch (IllegalThreadStateException ex)
            {
                System.out.println(parentGroup.getName() +
                    " is already empty: " +
                    ex.toString());
            }
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Thread-0 has finished executing.
Thread-1 has finished executing.
Child ThreadGroup is already empty: java.lang.IllegalThreadStateException: The thread has b
een destroyed or the group is empty
Parent ThreadGroup is already empty: java.lang.IllegalThreadStateException: The thread has
been destroyed or the group is empty
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.list()

Displays a listing of the 290ee7fa-aa87-4288-8644-5b2d5e6a3817, including the name and maximum priority of the **290ee7fa-aa87-4288-8644-5b2d5e6a3817**, as well as the names and priorities of the threads in the **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

J#

```
public void list()
```

Example

```
// threadgroup_list.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // List the contents of the ThreadGroups.
            System.out.println("\nListing for ThreadGroup " +
                parentGroup.getName() + ":");
            parentGroup.list();

            System.out.println("\nListing for ThreadGroup " +
                childGroup.getName() + ":");
            childGroup.list();

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
```

```
            {
                System.out.println(ex.toString());
            }
        }

        // Implements Runnable.run()
        public void run()
        {
            for (int i = 0; i < 100000000; i++)
            {
                // Just increment a counter.
                counter++;
            }

            System.out.println(Thread.currentThread().getName() +
                " has finished executing.");
        }

        private int counter = 0;
    }

    /*
    Sample Output:
    Starting Thread-0...
    Starting Thread-1...
    Active threads in group "Parent ThreadGroup" = 2
    Thread-0 has finished executing.
    Thread-1 has finished executing.

    Listing for ThreadGroup Parent ThreadGroup:
    java.lang.ThreadGroup[name=Parent ThreadGroup,maxpri=10]
        Thread[Thread-0,5,Parent ThreadGroup]
        java.lang.ThreadGroup[name=Child ThreadGroup,maxpri=10]
            Thread[Thread-1,5,Child ThreadGroup]

    Listing for ThreadGroup Child ThreadGroup:
    java.lang.ThreadGroup[name=Child ThreadGroup,maxpri=10]
        Thread[Thread-1,5,Child ThreadGroup]
    */
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.parentOf(ThreadGroup group)

Determines whether the current 290ee7fa-aa87-4288-8644-5b2d5e6a3817 is the parent of the given **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

J#

```
public final boolean parentOf(java.lang.ThreadGroup group)
```

Parameters

Parameter	Description
group	A ThreadGroup object to determine if it is a child of the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Property Value/Return Value

true if the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817** is the parent of the given **290ee7fa-aa87-4288-8644-5b2d5e6a3817**; **false** otherwise.

Example

```
// threadgroup_parentof.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Determine which ThreadGroup is the parent and
            // which is the child.
            boolean isParent = parentGroup.parentOf(childGroup);
            System.out.println(parentGroup.getName() +
                " is the parent of " + childGroup.getName() +
```

```
                "? " + isParent);

            isParent = childGroup.parentOf(parentGroup);
            System.out.println(childGroup.getName() +
                " is the parent of " + parentGroup.getName() +
                "? " + isParent);

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-0 has finished executing.
Parent ThreadGroup is the parent of Child ThreadGroup? true
Thread-1 has finished executing.
Child ThreadGroup is the parent of Parent ThreadGroup? false
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.resume()

Resumes execution of all threads in the 290ee7fa-aa87-4288-8644-5b2d5e6a3817 that have been suspended.

J#

```
public final void resume()
```

Remarks

Note For Visual J# 2005, the permission sets for this method have changed. Whereas no permission was needed in earlier versions, now permission is demanded when calling this method. See SecurityAttribute and SecurityAction for more information.

Example

```
// threadgroup_resume.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Suspend all threads in parentGroup.
            System.out.println("Suspending all threads in " +
                parentGroup.getName() + "...");
            parentGroup.suspend();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Block until the other threads finish or for 1000 ms,
            // whichever comes first.
            thr2.join(1000);
            thr3.join();
```

```
            // Resume all thread in parentGroup.
            System.out.println("Resuming all threads in " +
                parentGroup.getName());
            parentGroup.resume();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Suspending all threads in Parent ThreadGroup...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-1 has finished executing.
Resuming all threads in Parent ThreadGroup
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.setDaemon(boolean b)

Sets a value indicating whether the current 290ee7fa-aa87-4288-8644-5b2d5e6a3817 is a daemon (background) **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

J#

```
public final void setDaemon(boolean b)
```

Parameters

Parameter	Description
b	**true** to make the current **290ee7fa-aa87-4288-8644-5b2d5e6a3817** a daemon (background) **290ee7fa-aa87-4288-8644-5b2d5e6a3817**; **false** otherwise.

Example

```
// threadgroup_setdaemon.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            parentGroup.setDaemon(false);

            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");
            childGroup.setDaemon(true);

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Display which ThreadGroups are daemon ThreadGroups.
            System.out.println(parentGroup.getName() +
                " is a daemon ThreadGroup? " +
                parentGroup.isDaemon());
            System.out.println(childGroup.getName() +
```

```
                " is a daemon ThreadGroup? " +
                childGroup.isDaemon());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-0 has finished executing.
Parent ThreadGroup is a daemon ThreadGroup? false
Child ThreadGroup is a daemon ThreadGroup? true
Thread-1 has finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.setMaxPriority(int pri)

Sets the maximum priority that can be set for any thread in the 290ee7fa-aa87-4288-8644-5b2d5e6a3817.

J#

```
public final void setMaxPriority(int pri)
```

Parameters

Parameter	Description
pri	A value indicating the maximum priority that can be set for any thread in the **290ee7fa-aa87-4288-8644-5b2d5e6a3817**.

Remarks

If an attempt is made to add a thread with a higher priority than the value set in setMaxPriority to a **290ee7fa-aa87-4288-8644-5b2d5e6a3817**, then the priority for that thread will be lowered to the value set in setMaxPriority.

The default priority for a thread is defined by 31c5b51c-29cd-4f46-a943-6275f0acf0a2. The maximum priority for a thread is defined by 956ddc7b-7eb0-4857-a442-fd0d6d2d88b4. The minimum priority for a thread is defined by f8387b47-a01d-4f34-8d67-10e44c6c24e6.

Example

```
// threadgroup_setmaxpriority.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.  Set the maximum priority for
            // the parent to 8 and for the child to 5.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            parentGroup.setMaxPriority(Thread.MAX_PRIORITY - 2);

            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");
            childGroup.setMaxPriority(Thread.NORM_PRIORITY);

            // Create a second thread with maximum priority (pri 10)
            // and start it.
            Thread thr2 = new Thread(parentGroup, this);
            thr2.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread with maximum priority (pri 10)
            // and start it.
            Thread thr3 = new Thread(childGroup, this);
            thr3.setPriority(Thread.MAX_PRIORITY);
            System.out.println("Starting " +
                thr3.getName() + "...");
```

```
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        // Display what the priority was actually set to when the
        // thread was added to the ThreadGroup.
        System.out.println(Thread.currentThread().getName() +
            " [priority = " + Thread.currentThread().getPriority() +
            "] finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-0 [priority = 8] finished executing.
Thread-1 [priority = 5] finished executing.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.stop()

Stops execution of all threads in the 290ee7fa-aa87-4288-8644-5b2d5e6a3817.

J#

```
public final void stop()
```

Remarks

Note For Visual J# 2005, the permission sets for this method have changed. Whereas no permission was needed in earlier versions, now permission is demanded when calling this method. See SecurityAttribute and SecurityAction for more information.

Example

```
// threadgroup_stop.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        // Create a new ThreadGroup and a child for that
        // new ThreadGroup.
        ThreadGroup parentGroup =
            new ThreadGroup("Parent ThreadGroup");
        ThreadGroup childGroup =
            new ThreadGroup(parentGroup, "Child ThreadGroup");

        // Create a second thread and start it.
        Thread thr2 = new Thread(parentGroup, this);
        System.out.println("Starting " +
            thr2.getName() + "...");
        thr2.start();

        // Create a third thread and start it.
        Thread thr3 = new Thread(childGroup, this);
        System.out.println("Starting " +
            thr3.getName() + "...");
        thr3.start();

        // Display the number of active threads in the
        // new ThreadGroup.
        System.out.println("Active threads in group \"" +
            parentGroup.getName() + "\" = " +
            parentGroup.activeCount());

        // The threads are taking a long time.
        childGroup.stop();
        System.out.println(childGroup.getName() +
            " has been manually stopped.");

        parentGroup.stop();
        System.out.println(parentGroup.getName() +
            " has been manually stopped.");
    }

    // Implements Runnable.run()
    public void run()
```

```
    {
        for (int i = 0; i < Integer.MAX_VALUE; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Child ThreadGroup has been manually stopped.
Parent ThreadGroup has been manually stopped.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.suspend()

Suspends execution of all threads in the 290ee7fa-aa87-4288-8644-5b2d5e6a3817 until resume is called.

J#

```
public final void suspend()
```

Remarks

Note For Visual J# 2005, the permission sets for this method have changed. Whereas no permission was needed in earlier versions, now permission is demanded when calling this method. See SecurityAttribute and SecurityAction for more information.

Example

```
// threadgroup_suspend.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new ThreadGroup and a child for that
            // new ThreadGroup.
            ThreadGroup parentGroup =
                new ThreadGroup("Parent ThreadGroup");
            ThreadGroup childGroup =
                new ThreadGroup(parentGroup, "Child ThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Suspend all threads in parentGroup.
            System.out.println("Suspending all threads in " +
                parentGroup.getName() + "...");
            parentGroup.suspend();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            // Display the number of active threads in the
            // new ThreadGroup.
            System.out.println("Active threads in group \"" +
                parentGroup.getName() + "\" = " +
                parentGroup.activeCount());

            // Block until the other threads finish or for 1000 ms,
            // whichever comes first.
            thr2.join(1000);
            thr3.join();
```

```
            // Resume all thread in parentGroup.
            System.out.println("Resuming all threads in " +
                parentGroup.getName());
            parentGroup.resume();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        for (int i = 0; i < 100000000; i++)
        {
            // Just increment a counter.
            counter++;
        }

        System.out.println(Thread.currentThread().getName() +
            " has finished executing.");
    }

    private int counter = 0;
}

/*
Sample Output:
Starting Thread-0...
Suspending all threads in Parent ThreadGroup...
Starting Thread-1...
Active threads in group "Parent ThreadGroup" = 2
Thread-1 has finished executing.
Resuming all threads in Parent ThreadGroup
*/
```

See Also

Reference

java.lang.ThreadGroup

java.lang.ThreadGroup.uncaughtException(Thread thrd, Throwable e)

Prints the stack trace for any uncaught exceptions.

J#

```
public void uncaughtException(java.lang.Thread thrd, java.lang.Throwable e)
```

Parameters

Parameter	Description
thrd	The thread where the uncaught exception was thrown.
e	The exception that was thrown and not caught.

Remarks

This method is used to handle any unhandled exceptions from threads in the thread group. When any thread in a thread group leaves an exception unhandled, this methods will be called with the thread object and the exception -- giving the thread group a chance to handle it. This method receives all unhandled exceptions and errors including 2d622dab-0fb9-4756-8474-b94e83c9aae1Death -- which is thrown when a thread is destroyed.

Example

```
// threadgroup_uncaughtexception.jsl

public class Program implements Runnable
{
    public static void main(String[] args)
    {
        Program p = new Program();
        p.init();
    }

    public void init()
    {
        try
        {
            // Create a new MyThreadGroup (code below) and a child
            // for that new MyThreadGroup.
            MyThreadGroup parentGroup =
                new MyThreadGroup("Parent MyThreadGroup");
            MyThreadGroup childGroup =
                new MyThreadGroup(parentGroup, "Child MyThreadGroup");

            // Create a second thread and start it.
            Thread thr2 = new Thread(parentGroup, this);
            System.out.println("Starting " +
                thr2.getName() + "...");
            thr2.start();

            // Create a third thread and start it.
            Thread thr3 = new Thread(childGroup, this);
            System.out.println("Starting " +
                thr3.getName() + "...");
            thr3.start();

            try
            {
                // Give some time for the child threads to start.
                Thread.sleep(500);
            }
```

```
            catch (InterruptedException ex)
            {
                // Ignore this exception here
            }

            // Interrupt the two sub-threads.
            thr2.interrupt();
            thr3.interrupt();

            // Block until the other threads finish.
            thr2.join();
            thr3.join();
        }
        catch (InterruptedException ex)
        {
            System.out.println(ex.toString());
        }
    }

    // Implements Runnable.run()
    public void run()
    {
        try
        {
            System.out.println(Thread.currentThread().getName() +
                " is executing.");

            while (true)
            {
                Thread.sleep(500);
            }
        }
        catch (InterruptedException ex)
        {
            Thread currThread = Thread.currentThread();
            System.out.println(currThread.getName() +
                " was interrupted: " + ex.toString());

            // Rethrow the exception as a RuntimeException.
            // This will be sent to the ThreadGroup.
            throw new RuntimeException(ex.getMessage());

        }
    }
}

class MyThreadGroup extends ThreadGroup
{
    // Constructor requiring a name for the ThreadGroup.
    MyThreadGroup(String name)
    {
       super(name);
    }

    // Constructor requiring the parent and a name for the
    // ThreadGroup.
    MyThreadGroup(ThreadGroup parent, String name)
    {
        super(parent, name);
    }

    public void uncaughtException(Thread th, Throwable t)
    {
        System.out.println(th + " has unhandled exception: " + t);
    }

    // Hide default constructor.
    private MyThreadGroup()
```

```
    {
    }
}



/*
Starting Thread-0...
Starting Thread-1...
Thread-0 is executing.
Thread-1 is executing.
Thread-0 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
Thread-1 was interrupted: java.lang.InterruptedException: Thread has been interrupted from
a waiting state.
Thread[Thread-0,5,Parent MyThreadGroup] has unhandled exception: java.lang.RuntimeException
: Thread has been interrupted from a waiting state.
*/
```

See Also

Reference

java.lang.ThreadGroup

java.security.acl

Contains classes and interfaces used for processing Access Control Lists to grant or deny permissions to individuals or groups of individuals.

Classes

Class	Description
AclNotFoundException	The exception that is thrown when attempting to modify or view an Access Control List that cannot be found.
LastOwnerException	The exception that is thrown when attempting to remove the last owner from an Access Control List.
NotOwnerException	The exception that is thrown when attempting to modify an Access Control List that is not owned by the caller.

Interfaces

Interface	Description
Acl	Represents an Access Control List used to explicitly grant or deny permissions to a file or folder to individuals or groups of individuals.
AclEntry	Represents an entry in an Access Control List.
Group	Represents a group of individuals within an Access Control List. All members of the group can be granted or denied permission by granting or denying the permission to the group.
Owner	Represents an owner of an Access Control List.
Permission	Represents a permission within an entry in the Access Control List.

Acl Interface

Represents an Access Control List used to explicitly grant or deny permissions to a file or folder to individuals or groups of individuals.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public interface java.security.acl.Acl
    extends java.security.acl.Owner
```

See Also

Concepts

Acl Members

java.security.acl Package

Acl Members

Represents an Access Control List used to explicitly grant or deny permissions to a file or folder to individuals or groups of individuals.

The following tables list the members exposed by the Acl type.

Public Methods

Name	Description
addEntry	Adds an entry to an Access Control List.
checkPermission	Determines whether an individual has a certain permission granted in the Access Control List.
entries	Returns an Enumeration object that can be used to enumerate over the AclEntry objects in the Access Control List.
getName	Gets the name of the Access Control List.
getPermissions	Returns an Enumeration object that can be used to enumerate over the Permission objects for a given user in the Access Control List.
removeEntry	Removes an entry from an Access Control List.
setName	Sets the name of the Access Control List.
toString	Displays a human-readable representation of an Acl object.

See Also

Reference

Acl Interface

Concepts

java.security.acl Package

Acl Methods

Public Methods

Name	Description
addEntry	Adds an entry to an Access Control List.
checkPermission	Determines whether an individual has a certain permission granted in the Access Control List.
entries	Returns an Enumeration object that can be used to enumerate over the AclEntry objects in the Access Control List.
getName	Gets the name of the Access Control List.
getPermissions	Returns an Enumeration object that can be used to enumerate over the Permission objects for a given user in the Access Control List.
removeEntry	Removes an entry from an Access Control List.
setName	Sets the name of the Access Control List.
toString	Displays a human-readable representation of an Acl object.

See Also

Reference

Acl Interface

Concepts

java.security.acl Package

Acl.addEntry Method

Adds an entry to an Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean addEntry(
    java.security.Principal caller,
    java.security.acl.AclEntry entry) throws java.security.acl.NotOwnerException;
```

Parameters

caller

The person, represented by a **Principal** object, attempting to add the entry to the Access Control List.

entry

The entry, in the form of an AclEntry object, to add to the Access Control List.

Return Value

true if the entry was successfully added to the Access Control List; false otherwise.

See Also

Reference

Acl Interface

Concepts

Acl Members

java.security.acl Package

Acl.checkPermission Method

Determines whether an individual has a certain permission granted in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean checkPermission(
    java.security.Principal principal,
    java.security.acl.Permission permission);
```

Parameters

principal

The person, represented by a **Principal** object, whose permissions are being checked.

permission

The permission, in the form of a Permission object, to check.

Return Value

true if the principal has the given permission; false otherwise.

See Also

Reference

Acl Interface

Concepts

Acl Members

java.security.acl Package

Acl.entries Method

Returns an Enumeration object that can be used to enumerate over the AclEntry objects in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Enumeration entries();
```

Return Value

An Enumeration object that can be used to enumerate over the AclEntry objects in the Access Control List.

See Also

Reference

Acl Interface

Concepts

Acl Members

java.security.acl Package

Acl.getName Method

Gets the name of the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getName();
```

Return Value

The name of the Access Control List.

See Also

Reference

Acl Interface

Concepts

Acl Members

java.security.acl Package

Acl.getPermissions Method

Returns an Enumeration object that can be used to enumerate over the Permission objects for a given user in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Enumeration getPermissions(
    java.security.Principal user);
```

Parameters

user

The person, represented by a **Principal** object, whose permissions are being returned.

Return Value

An Enumeration object that can be used to enumerate over the Permission objects for a given user in the Access Control List.

See Also

Reference

Acl Interface

Concepts

Acl Members

java.security.acl Package

Acl.removeEntry Method

Removes an entry from an Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean removeEntry(
    java.security.Principal caller,
    java.security.acl.AclEntry entry) throws java.security.acl.NotOwnerException;
```

Parameters

caller

The person, represented by a **Principal** object, attempting to remove the entry from the Access Control List.

entry

The entry, in the form of an AclEntry object, to remove from the Access Control List.

Return Value

true if the entry was successfully removed from the Access Control List; false otherwise.

See Also

Reference

Acl Interface

Concepts

Acl Members

java.security.acl Package

Acl.setName Method

Sets the name of the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract void setName(
    java.security.Principal caller,
    java.lang.String name) throws java.security.acl.NotOwnerException;
```

Parameters

caller

The person, represented by a **Principal** object, attempting to set the name of the Access Control List.

name

The name of the Access Control List.

See Also

Reference

Acl Interface

Concepts

Acl Members

java.security.acl Package

Acl.toString Method

Displays a human-readable representation of an Acl object.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String toString();
```

Return Value

A human-readable representation of an Acl object.

See Also

Reference

Acl Interface

Concepts

Acl Members

java.security.acl Package

AclEntry Interface

Represents an entry in an Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public interface java.security.acl.AclEntry
    extends java.lang.Cloneable
```

See Also

Concepts

AclEntry Members

java.security.acl Package

AclEntry Members

Represents an entry in an Access Control List.

The following tables list the members exposed by the AclEntry type.

Public Methods

Name	Description
addPermission	Adds a permission to an entry in the Access Control List.
checkPermission	Determines whether a permissions is granted for an entry in the Access Control List.
getPrincipal	Gets the **Principal** object associated with an entry in the Access Control List.
isNegative	Determines whether the permissions in this entry are negative. A negative entry means that the permissions are denied rather than granted.
clone	Creates an instance of an AclEntry object that is a shallow copy of the current AclEntry object.
permissions	Returns an Enumeration object that can be used to enumerate over the Permission objects for an entry in the Access Control List.
removePermission	Removes a permission from an entry in the Access Control List.
setNegativePermissions	Sets the permissions in this entry to negative. A negative entry means that the permissions are denied rather than granted.
setPrincipal	Sets the Principal object associated with an entry in the Access Control List.
toString	Displays a human-readable representation of an AclEntry object.

See Also

Reference

AclEntry Interface

Concepts

java.security.acl Package

AclEntry Methods

Public Methods

Name	Description
addPermission	Adds a permission to an entry in the Access Control List.
checkPermission	Determines whether a permissions is granted for an entry in the Access Control List.
getPrincipal	Gets the **Principal** object associated with an entry in the Access Control List.
isNegative	Determines whether the permissions in this entry are negative. A negative entry means that the permissions are denied rather than granted.
clone	Creates an instance of an AclEntry object that is a shallow copy of the current AclEntry object.
permissions	Returns an Enumeration object that can be used to enumerate over the Permission objects for an entry in the Access Control List.
removePermission	Removes a permission from an entry in the Access Control List.
setNegativePermissions	Sets the permissions in this entry to negative. A negative entry means that the permissions are denied rather than granted.
setPrincipal	Sets the Principal object associated with an entry in the Access Control List.
toString	Displays a human-readable representation of an AclEntry object.

See Also

Reference

AclEntry Interface

Concepts

java.security.acl Package

AclEntry.addPermission Method

Adds a permission to an entry in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean addPermission(
    java.security.acl.Permission permission);
```

Parameters

permission

The permission, in the form of a Permission object, to add to an entry in the Access Control List.

Return Value

true if the permission was successfully added; false otherwise.

See Also

Reference

AclEntry Interface

Concepts

AclEntry Members

java.security.acl Package

AclEntry.checkPermission Method

Determines whether a permissions is granted for an entry in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean checkPermission(
    java.security.acl.Permission permission);
```

Parameters

permission

The permission, in the form of a Permission object, to check.

Return Value

true if the given permission is granted for the current entry in the Access Control List; false otherwise.

See Also

Reference

AclEntry Interface

Concepts

AclEntry Members

java.security.acl Package

AclEntry.clone Method

Creates an instance of an AclEntry object that is a shallow copy of the current AclEntry object.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object clone();
```

Return Value

A shallow copy of the current AclEntry object.

See Also

Reference

AclEntry Interface

Concepts

AclEntry Members

java.security.acl Package

AclEntry.getPrincipal Method

Gets the **Principal** object associated with an entry in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.security.Principal getPrincipal();
```

Return Value

The Principal object associated with an entry in the Access Control List.

See Also

Reference

AclEntry Interface

Concepts

AclEntry Members

java.security.acl Package

AclEntry.isNegative Method

Determines whether the permissions in this entry are negative. A negative entry means that the permissions are denied rather than granted.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isNegative();
```

Return Value

true if the entry is negative; false otherwise.

See Also

Reference

AclEntry Interface

Concepts

AclEntry Members

java.security.acl Package

AclEntry.permissions Method

Returns an Enumeration object that can be used to enumerate over the Permission objects for an entry in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Enumeration permissions();
```

Return Value

An Enumeration object that can be used to enumerate over the Permission objects for an entry in the Access Control List.

See Also

Reference

AclEntry Interface

Concepts

AclEntry Members

java.security.acl Package

AclEntry.removePermission Method

Removes a permission from an entry in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean removePermission(
    java.security.acl.Permission permission);
```

Parameters

permission

The permission, in the form of a Permission object, to remove from an entry in the Access Control List.

Return Value

true if the permission was successfully removed; false otherwise.

See Also

Reference

AclEntry Interface

Concepts

AclEntry Members

java.security.acl Package

AclEntry.setNegativePermissions Method

Sets the permissions in this entry to negative. A negative entry means that the permissions are denied rather than granted.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract void setNegativePermissions();
```

See Also

Reference

AclEntry Interface

Concepts

AclEntry Members

java.security.acl Package

AclEntry.setPrincipal Method

Sets the Principal object associated with an entry in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean setPrincipal(
    java.security.Principal user);
```

Parameters

user

The Principal object associated with an entry in the Access Control List.

Return Value

true if the principal for the current entry in the Access Control List was successfully set; false otherwise.

See Also

Concepts

AclEntry Members

java.security.acl Package

Other Resources

AclEntry Interface

AclEntry.toString Method

Displays a human-readable representation of an AclEntry object.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String toString();
```

Return Value

A human-readable representation of an AclEntry object.

See Also

Reference

AclEntry Interface

Concepts

AclEntry Members

java.security.acl Package

AclNotFoundException Class

The exception that is thrown when attempting to modify or view an Access Control List that cannot be found.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public class java.security.acl.AclNotFoundException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.security.acl.AclNotFoundException

See Also

Concepts

AclNotFoundException Members

java.security.acl Package

AclNotFoundException Members

The exception that is thrown when attempting to modify or view an Access Control List that cannot be found.

The following tables list the members exposed by the AclNotFoundException type.

Public Constructors

Name	Description
AclNotFoundException	Overloaded. Initializes a new instance of an AclNotFoundException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AclNotFoundException Class

Concepts

java.security.acl Package

AclNotFoundException Constructor

Initializes a new instance of an AclNotFoundException object.

Overload List

Name	Description
AclNotFoundException ()	Initializes a new instance of an AclNotFoundException object.
AclNotFoundException (SerializationInfo, StreamingContext)	Initializes a new instance of an AclNotFoundException object duri ng deserialization.
AclNotFoundException (String, Exception)	Initializes a new instance of an AclNotFoundException with the gi ven message and inner exception.

See Also

Reference

AclNotFoundException Class

Concepts

AclNotFoundException Members

java.security.acl Package

AclNotFoundException Constructor ()

Initializes a new instance of an AclNotFoundException object.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public java.security.acl.AclNotFoundException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

AclNotFoundException Class

Concepts

AclNotFoundException Members

java.security.acl Package

AclNotFoundException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an AclNotFoundException object during deserialization.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
protected java.security.acl.AclNotFoundException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
AclNotFoundException Class
Concepts
AclNotFoundException Members
java.security.acl Package

AclNotFoundException Constructor (String, Exception)

Initializes a new instance of an AclNotFoundException with the given message and inner exception.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public java.security.acl.AclNotFoundException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the AclNotFoundException being thrown.

See Also

Reference

AclNotFoundException Class

Concepts

AclNotFoundException Members

java.security.acl Package

AclNotFoundException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AclNotFoundException Class

Concepts

java.security.acl Package

AclNotFoundException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

AclNotFoundException Class

Concepts

java.security.acl Package

Group Interface

Represents a group of individuals within an Access Control List. All members of the group can be granted or denied permission by granting or denying the permission to the group.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public interface java.security.acl.Group
    extends java.security.Principal
```

See Also

Concepts

Group Members

java.security.acl Package

Group Members

Represents a group of individuals within an Access Control List. All members of the group can be granted or denied permission by granting or denying the permission to the group.

The following tables list the members exposed by the Group type.

Public Methods

Name	Description
addMember	Adds a member to the group.
isMember	Determines whether an individual is a member of the group.
members	Returns an Enumeration object that can be used to enumerate over the members of the group.
removeMember	Removes a member from the group.

See Also

Reference

Group Interface

Concepts

java.security.acl Package

Group Methods

Public Methods

Name	Description
addMember	Adds a member to the group.
isMember	Determines whether an individual is a member of the group.
members	Returns an Enumeration object that can be used to enumerate over the members of the group.
removeMember	Removes a member from the group.

See Also

Reference

Group Interface

Concepts

java.security.acl Package

Group.addMember Method

Adds a member to the group.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean addMember(
    java.security.Principal user);
```

Parameters

user

The person, represented by a **Principal** object, to add to the group.

Return Value

true if the user was successfully added to the group; false otherwise.

See Also

Reference

Group Interface

Concepts

Group Members

java.security.acl Package

Group.isMember Method

Determines whether an individual is a member of the group.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isMember(
    java.security.Principal member);
```

Parameters

member

The person, represented by a **Principal** object, whose membership is being checked.

Return Value

true if the individual is a member of the group; false otherwise.

See Also

Reference

Group Interface

Concepts

Group Members

java.security.acl Package

Group.members Method

Returns an Enumeration object that can be used to enumerate over the members of the group.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Enumeration members();
```

Return Value

An Enumeration object that can be used to enumerate over the members of the group.

See Also

Reference

Group Interface

Concepts

Group Members

java.security.acl Package

Group.removeMember Method

Removes a member from the group.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean removeMember(
    java.security.Principal user);
```

Parameters

user

The person, represented by a **Principal** object, to remove from the group.

Return Value

true if the user was successfully removed from the group; false otherwise.

See Also

Reference

Group Interface

Concepts

Group Members

java.security.acl Package

LastOwnerException Class

The exception that is thrown when attempting to remove the last owner from an Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public class java.security.acl.LastOwnerException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.security.acl.LastOwnerException

See Also

Concepts

LastOwnerException Members

java.security.acl Package

LastOwnerException Members

The exception that is thrown when attempting to remove the last owner from an Access Control List.

The following tables list the members exposed by the LastOwnerException type.

Public Constructors

Name	Description
LastOwnerException	Overloaded. Initializes a new instance of a LastOwnerException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

LastOwnerException Class

Concepts

java.security.acl Package

LastOwnerException Constructor

Initializes a new instance of a LastOwnerException object.

Overload List

Name	Description
LastOwnerException ()	Initializes a new instance of a LastOwnerException object.
LastOwnerException (SerializationInfo, StreamingContext)	Initializes a new instance of a LastOwnerException object during deserialization.
LastOwnerException (String, Exception)	Initializes a new instance of a LastOwnerException object during deserialization.

See Also

Reference

LastOwnerException Class

Concepts

LastOwnerException Members

java.security.acl Package

LastOwnerException Constructor ()

Initializes a new instance of a LastOwnerException object.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public java.security.acl.LastOwnerException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

LastOwnerException Class

Concepts

LastOwnerException Members

java.security.acl Package

LastOwnerException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a LastOwnerException object during deserialization.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
protected java.security.acl.LastOwnerException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

LastOwnerException Class

Concepts

LastOwnerException Members

java.security.acl Package

LastOwnerException Constructor (String, Exception)

Initializes a new instance of a LastOwnerException object during deserialization.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public java.security.acl.LastOwnerException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the LastOwnerException object being thrown.

See Also

Reference

LastOwnerException Class

Concepts

LastOwnerException Members

java.security.acl Package

LastOwnerException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

LastOwnerException Class

Concepts

java.security.acl Package

LastOwnerException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

LastOwnerException Class

Concepts

java.security.acl Package

NotOwnerException Class

The exception that is thrown when attempting to modify an Access Control List that is not owned by the caller.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public class java.security.acl.NotOwnerException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.security.acl.NotOwnerException

See Also

Concepts

NotOwnerException Members

java.security.acl Package

NotOwnerException Members

The exception that is thrown when attempting to modify an Access Control List that is not owned by the caller.

The following tables list the members exposed by the NotOwnerException type.

Public Constructors

Name	Description
NotOwnerException	Overloaded. Initializes a new instance of a NotOwnerException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NotOwnerException Class

Concepts

java.security.acl Package

NotOwnerException Constructor

Initializes a new instance of a NotOwnerException object.

Overload List

Name	Description
NotOwnerException ()	Initializes a new instance of a NotOwnerException object.
NotOwnerException (SerializationInfo, StreamingContext)	Initializes a new instance of a NotOwnerException object during deserialization.
NotOwnerException (String, Exception)	Initializes a new instance of a NotOwnerException object with the given message and inner exception.

See Also

Reference

NotOwnerException Class

Concepts

NotOwnerException Members

java.security.acl Package

NotOwnerException Constructor ()

Initializes a new instance of a NotOwnerException object.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public java.security.acl.NotOwnerException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NotOwnerException Class

Concepts

NotOwnerException Members

java.security.acl Package

NotOwnerException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a NotOwnerException object during deserialization.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
protected java.security.acl.NotOwnerException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
NotOwnerException Class
Concepts
NotOwnerException Members
java.security.acl Package

NotOwnerException Constructor (String, Exception)

Initializes a new instance of a NotOwnerException object with the given message and inner exception.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public java.security.acl.NotOwnerException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the NotOwnerException being thrown.

See Also

Reference

NotOwnerException Class

Concepts

NotOwnerException Members

java.security.acl Package

NotOwnerException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NotOwnerException Class

Concepts

java.security.acl Package

NotOwnerException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NotOwnerException Class

Concepts

java.security.acl Package

Owner Interface

Represents an owner of an Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public interface java.security.acl.Owner
```

See Also

Concepts

Owner Members

java.security.acl Package

Owner Members

Represents an owner of an Access Control List.

The following tables list the members exposed by the Owner type.

Public Methods

Name	Description
addOwner	Adds an owner to the Access Control List.
deleteOwner	Deletes an owner from the Access Control List.
isOwner	Determines whether an individual is an owner of an Access Control List.

See Also

Reference

Owner Interface

Concepts

java.security.acl Package

Owner Methods

Public Methods

Name	Description
addOwner	Adds an owner to the Access Control List.
deleteOwner	Deletes an owner from the Access Control List.
isOwner	Determines whether an individual is an owner of an Access Control List.

See Also

Reference

Owner Interface

Concepts

java.security.acl Package

Owner.addOwner Method

Adds an owner to the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean addOwner(
    java.security.Principal caller,
    java.security.Principal owner) throws java.security.acl.NotOwnerException;
```

Parameters

caller

The person, represented by a **Principal** object, attempting to add the owner to the Access Control List.

owner

The person, represented by a Principal object, to set as the owner of the Access Control List.

Return Value

true if the owner was successfully added; false otherwise.

See Also

Reference

Owner Interface

Concepts

Owner Members

java.security.acl Package

Owner.deleteOwner Method

Deletes an owner from the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean deleteOwner(
    java.security.Principal caller,
    java.security.Principal owner) throws java.security.acl.NotOwnerException, java.securit
y.acl.LastOwnerException;
```

Parameters

caller

The person, represented by a **Principal** object, attempting to delete the owner from the Access Control List.

owner

The person, represented by a **Principal** object, to delete as the owner of the Access Control List.

Return Value

true if the owner was successfully deleted; false otherwise.

See Also

Reference

Owner Interface

Concepts

Owner Members

java.security.acl Package

Owner.isOwner Method

Determines whether an individual is an owner of an Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isOwner(
    java.security.Principal owner);
```

Parameters

owner

The person, represented by a **Principal** object, whose ownership status is being determined.

Return Value

true if owner is an owner of the Access Control List; false otherwise.

See Also

Reference

Owner Interface

Concepts

Owner Members

java.security.acl Package

Permission Interface

Represents a permission within an entry in the Access Control List.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public interface java.security.acl.Permission
```

See Also

Concepts

Permission Members

Permission Methods

Permission Members

Represents a permission within an entry in the Access Control List.

The following tables list the members exposed by the Permission type.

Public Methods

Name	Description
equals	Determines whether two Permission objects are equal.
toString	Displays a human-readable representation of a Permission object.

See Also

Reference

Permission Interface

Concepts

java.security.acl Package

Permission Methods

Public Methods

Name	Description
equals	Determines whether two Permission objects are equal.
toString	Displays a human-readable representation of a Permission object.

See Also

Reference

Permission Interface

Concepts

java.security.acl Package

Permission.equals Method

Determines whether two Permission objects are equal.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean equals(
    java.lang.Object another);
```

Parameters

another

A Permission object to compare with the current Permission object for equality.

Return Value

true if the two Permission objects are equal; false otherwise.

See Also

Reference

Permission Interface

Concepts

Permission Members

java.security.acl Package

Permission.toString Method

Displays a human-readable representation of a Permission object.

Package: java.security.acl

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String toString();
```

Return Value

A human-readable representation of a Permission object.

See Also

Reference

Permission Interface

Concepts

Permission Members

java.security.acl Package

java.sql

Contains the classes and interfaces used for accessing and manipulating data stored in a relational database.

Interfaces

Interface	Description
Connection	Represents a connection to an external database such as SQL Server.

CallableStatement Interface

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public interface java.sql.CallableStatement
    extends java.sql.PreparedStatement
```

See Also

Concepts

CallableStatement Members

java.sql Package

CallableStatement Members

The following tables list the members exposed by the CallableStatement type.

Public Methods

Name	Description
getBigDecimal	
getBoolean	
getByte	
getBytes	
getDate	
getDouble	
getFloat	
getInt	
getLong	
getObject	
getShort	
getString	
getTime	
getTimestamp	
registerOutParameter	Overloaded.
wasNull	

See Also

Reference

CallableStatement Interface

Concepts

java.sql Package

CallableStatement Methods

Public Methods

Name	Description
getBigDecimal	
getBoolean	
getByte	
getBytes	
getDate	
getDouble	
getFloat	
getInt	
getLong	
getObject	
getShort	
getString	
getTime	
getTimestamp	
registerOutParameter	Overloaded.
wasNull	

See Also

Reference

CallableStatement Interface

Concepts

java.sql Package

CallableStatement.getBigDecimal Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.math.BigDecimal getBigDecimal(
    int parameterIndex,
    int scale) throws java.sql.SQLException;
```

Parameters

parameterIndex

scale

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getBoolean Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean getBoolean(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getByte Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract byte getByte(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getBytes Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract byte[] getBytes(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getDate Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Date getDate(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getDouble Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract double getDouble(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getFloat Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract float getFloat(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getInt Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getInt(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also
Reference
CallableStatement Interface
Concepts
CallableStatement Members
java.sql Package

CallableStatement.getLong Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract long getLong(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getObject Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object getObject(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getShort Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract short getShort(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getString Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getString(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also
Reference
CallableStatement Interface
Concepts
CallableStatement Members
java.sql Package

CallableStatement.getTime Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Time getTime(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.getTimestamp Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Timestamp getTimestamp(
    int parameterIndex) throws java.sql.SQLException;
```

Parameters

parameterIndex

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.registerOutParameter Method

Overload List

Name	Description
CallableStatement.registerOutParameter (int, int)	
CallableStatement.registerOutParameter (int, int, int)	

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.registerOutParameter Method (Int32, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void registerOutParameter(
    int parameterIndex,
    int sqlType) throws java.sql.SQLException;
```

Parameters

parameterIndex

sqlType

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

CallableStatement.registerOutParameter Method (Int32, Int32, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void registerOutParameter(
    int parameterIndex,
    int sqlType,
    int scale) throws java.sql.SQLException;
```

Parameters

parameterIndex

sqlType

scale

See Also
Reference
CallableStatement Interface
Concepts
CallableStatement Members
java.sql Package

CallableStatement.wasNull Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean wasNull() throws java.sql.SQLException;
```

See Also

Reference

CallableStatement Interface

Concepts

CallableStatement Members

java.sql Package

Connection Interface

Represents a connection to an external database such as SQL Server.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public interface java.sql.Connection
```

See Also

Concepts

Connection Members

java.sql Package

Connection Members

Represents a connection to an external database such as SQL Server.

The following tables list the members exposed by the Connection type.

Public Fields

Name	Description
TRANSACTION_NONE	If you do not set the transaction isolation or you set it to TRANSACTION_NONE, the transaction executes according to the default isolation level of the underlying ODBC driver.
TRANSACTION_READ_COMMITTED	Shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in non-repeatable reads or phantom data.
TRANSACTION_READ_UNCOMMITTED	A dirty read is possible, meaning that no shared locks are issued and no exclusive locks are honored.
TRANSACTION_REPEATABLE_READ	Locks are placed on all data that is used in a query, preventing other users from updating the data. Prevents non-repeatable reads but phantom rows are still possible.
TRANSACTION_SERIALIZABLE	A range lock is placed on the dataset, preventing other users from updating or inserting rows into the dataset until the transaction is complete.

Public Methods

Name	Description
clearWarnings	Clears all warnings reported by calls on this connection.
close	Closes the connection to the database.
commit	Commits all transactions called on the database since the last commit.
createStatement	Creates a new instance of a Statement object that represents a statement to call on the database.
getAutoCommit	Gets a value determining whether transactions are being automatically committed.
getCatalog	Gets the name of the database associated with this connection.
getMetaData	Gets the metadata associated with the database that this connection is using.
getTransactionIsolation	Gets a value representing the transaction locking behavior for this connection.
getWarnings	Gets all warnings reported by calls on this connection.
isClosed	Determines whether the connection to the database is closed.
isReadOnly	Determines whether the connection to the database is read-only.
nativeSQL	Gets the SQL command that will be passed to the database server, after any preprocessing by the database driver. The behavior of this method is database dependent.

prepareCall	Generates an instance of a CallableStatement object representing a SQL command that can be called o n the database.
prepareStatement	Generates an instance of a PreparedStatement object representing a SQL command that will likely be c alled on the database many times. This is a hint to the database server to optimize the statement by do ing a precompilation of the statement.
rollback	Undoes all transactions called on the database since the last commit operation.
setAutoCommit	Sets a value determining whether transactions are being automatically committed.
setCatalog	Sets the name of the database associated with this connection.
setReadOnly	Sets a value determining whether the connection to the database is read-only.
setTransactionIsolation	Sets a value representing the transaction locking behavior for this connection.

See Also

Reference

Connection Interface

Concepts

java.sql Package

Connection Fields

Public Fields

Name	Description
TRANSACTION_NONE	If you do not set the transaction isolation or you set it to TRANSACTION_NONE, the transaction executes according to the default isolation level of the underlying ODBC driver.
TRANSACTION_READ_COMMITTED	Shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in non-repeatable reads or phantom data.
TRANSACTION_READ_UNCOMMITTED	A dirty read is possible, meaning that no shared locks are issued and no exclusive locks are honored.
TRANSACTION_REPEATABLE_READ	Locks are placed on all data that is used in a query, preventing other users from updating the data. Prevents non-repeatable reads but phantom rows are still possible.
TRANSACTION_SERIALIZABLE	A range lock is placed on the dataset, preventing other users from updating or inserting rows into the dataset until the transaction is complete.

See Also

Reference

Connection Interface

Concepts

java.sql Package

Connection.TRANSACTION_NONE Field

If you do not set the transaction isolation or you set it to TRANSACTION_NONE, the transaction executes according to the default isolation level of the underlying ODBC driver.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int TRANSACTION_NONE;
```

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.TRANSACTION_READ_COMMITTED Field

Shared locks are held while the data is being read to avoid dirty reads, but the data can be changed before the end of the transaction, resulting in non-repeatable reads or phantom data.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int TRANSACTION_READ_COMMITTED;
```

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.TRANSACTION_READ_UNCOMMITTED Field

A dirty read is possible, meaning that no shared locks are issued and no exclusive locks are honored.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int TRANSACTION_READ_UNCOMMITTED;
```

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.TRANSACTION_REPEATABLE_READ Field

Locks are placed on all data that is used in a query, preventing other users from updating the data. Prevents non-repeatable reads but phantom rows are still possible.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int TRANSACTION_REPEATABLE_READ;
```

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.TRANSACTION_SERIALIZABLE Field

A range lock is placed on the dataset, preventing other users from updating or inserting rows into the dataset until the transaction is complete.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int TRANSACTION_SERIALIZABLE;
```

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection Methods

Public Methods

Name	Description
clearWarnings	Clears all warnings reported by calls on this connection.
close	Closes the connection to the database.
commit	Commits all transactions called on the database since the last commit.
createStatement	Creates a new instance of a Statement object that represents a statement to call on the database.
getAutoCommit	Gets a value determining whether transactions are being automatically committed.
getCatalog	Gets the name of the database associated with this connection.
getMetaData	Gets the metadata associated with the database that this connection is using.
getTransactionIsolation	Gets a value representing the transaction locking behavior for this connection.
getWarnings	Gets all warnings reported by calls on this connection.
isClosed	Determines whether the connection to the database is closed.
isReadOnly	Determines whether the connection to the database is read-only.
nativeSQL	Gets the SQL command that will be passed to the database server, after any preprocessing by the data base driver. The behavior of this method is database dependent.
prepareCall	Generates an instance of a CallableStatement object representing a SQL command that can be called o n the database.
prepareStatement	Generates an instance of a PreparedStatement object representing a SQL command that will likely be c alled on the database many times. This is a hint to the database server to optimize the statement by do ing a precompilation of the statement.
rollback	Undoes all transactions called on the database since the last commit operation.
setAutoCommit	Sets a value determining whether transactions are being automatically committed.
setCatalog	Sets the name of the database associated with this connection.
setReadOnly	Sets a value determining whether the connection to the database is read-only.
setTransactionIsolation	Sets a value representing the transaction locking behavior for this connection.

See Also

Reference

Connection Interface

Concepts

java.sql Package

Connection.clearWarnings Method

Clears all warnings reported by calls on this connection.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void clearWarnings() throws java.sql.SQLException;
```

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.close Method

Closes the connection to the database.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void close() throws java.sql.SQLException;
```

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.commit Method

Commits all transactions called on the database since the last commit.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void commit() throws java.sql.SQLException;
```

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.createStatement Method

Creates a new instance of a Statement object that represents a statement to call on the database.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Statement createStatement() throws java.sql.SQLException;
```

Return Value

A new instance of a Statement object that represents a statement to call on the database.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.getAutoCommit Method

Gets a value determining whether transactions are being automatically committed.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean getAutoCommit() throws java.sql.SQLException;
```

Return Value

true if transactions are being automatically committed; false otherwise.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.getCatalog Method

Gets the name of the database associated with this connection.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getCatalog() throws java.sql.SQLException;
```

Return Value

The name of the database associated with this connection.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.getMetaData Method

Gets the metadata associated with the database that this connection is using.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.DatabaseMetaData getMetaData() throws java.sql.SQLException;
```

Return Value

An instance of a DatabaseMetaData object that represents the metadata associated with the database that this connection is using.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.getTransactionIsolation Method

Gets a value representing the transaction locking behavior for this connection.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getTransactionIsolation() throws java.sql.SQLException;
```

Return Value

A value representing the transaction locking behavior for this connection. Valid values are TRANSACTION_NONE, TRANSACTION_READ_UNCOMMITTED, TRANSACTION_READ_COMMITTED, TRANSACTION_REPEATABLE_READ, and TRANSACTION_SERIALIZABLE.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.getWarnings Method

Gets all warnings reported by calls on this connection.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.SQLWarning getWarnings() throws java.sql.SQLException;
```

Return Value

An instance of a SQLWarning object containing all warnings reported by calls on this connection.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.isClosed Method

Determines whether the connection to the database is closed.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isClosed() throws java.sql.SQLException;
```

Return Value

true if the connection to the database is closed; false otherwise.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.isReadOnly Method

Determines whether the connection to the database is read-only.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isReadOnly() throws java.sql.SQLException;
```

Return Value

true if the connection to the database is read-only; false otherwise.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.nativeSQL Method

Gets the SQL command that will be passed to the database server, after any preprocessing by the database driver. The behavior of this method is database dependent.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String nativeSQL(
    java.lang.String sql) throws java.sql.SQLException;
```

Parameters

sql

The SQL command passed to the database driver.

Return Value

The SQL command that will be passed to the database server, after any preprocessing.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.prepareCall Method

Generates an instance of a CallableStatement object representing a SQL command that can be called on the database.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.CallableStatement prepareCall(
    java.lang.String sql) throws java.sql.SQLException;
```

Parameters

sql

The SQL command to call on the database.

Return Value

An instance of a CallableStatement object representing a SQL command that can be called on the database.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.prepareStatement Method

Generates an instance of a PreparedStatement object representing a SQL command that will likely be called on the database many times. This is a hint to the database server to optimize the statement by doing a precompilation of the statement.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.PreparedStatement prepareStatement(
    java.lang.String sql) throws java.sql.SQLException;
```

Parameters

sql

The SQL command to call on the database.

Return Value

An instance of a PreparedStatement object representing a SQL command that will likely be called on the database many times.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.rollback Method

Undoes all transactions called on the database since the last commit operation.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void rollback() throws java.sql.SQLException;
```

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.setAutoCommit Method

Sets a value determining whether transactions are being automatically committed.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setAutoCommit(
    boolean autoCommit) throws java.sql.SQLException;
```

Parameters

autoCommit

true to automatically commit transactions; false otherwise.

See Also
Reference
Connection Interface
Concepts
Connection Members
java.sql Package

Connection.setCatalog Method

Sets the name of the database associated with this connection.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setCatalog(
    java.lang.String catalog) throws java.sql.SQLException;
```

Parameters

catalog

The name of the database associated with this connection.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.setReadOnly Method

Sets a value determining whether the connection to the database is read-only.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setReadOnly(
    boolean readOnly) throws java.sql.SQLException;
```

Parameters

readOnly

true to make the connection to the database read-only; false otherwise.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

Connection.setTransactionIsolation Method

Sets a value representing the transaction locking behavior for this connection.

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setTransactionIsolation(
    int transactionIsolation) throws java.sql.SQLException;
```

Parameters

transactionIsolation

A value representing the transaction locking behavior for this connection. Valid values are TRANSACTION_NONE, TRANSACTION_READ_UNCOMMITTED, TRANSACTION_READ_COMMITTED, TRANSACTION_REPEATABLE_READ, and TRANSACTION_SERIALIZABLE.

See Also

Reference

Connection Interface

Concepts

Connection Members

java.sql Package

DatabaseMetaData Interface

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public interface java.sql.DatabaseMetaData
```

See Also

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData Members

The following tables list the members exposed by the DatabaseMetaData type.

Public Fields

Name	Description
bestRowNotPseudo	
bestRowPseudo	
bestRowSession	
bestRowTemporary	
bestRowTransaction	
bestRowUnknown	
columnNoNulls	
columnNullable	
columnNullableUnknown	
importedKeyCascade	
importedKeyInitiallyDeferred	
importedKeyInitiallyImmediate	
importedKeyNoAction	
importedKeyNotDeferrable	
importedKeyRestrict	
importedKeySetDefault	
importedKeySetNull	
procedureColumnIn	
procedureColumnInOut	
procedureColumnOut	
procedureColumnResult	
procedureColumnReturn	
procedureColumnUnknown	

procedureNoNulls	
procedureNoResult	
procedureNullable	
procedureNullableUnknown	
procedureResultUnknown	
procedureReturnsResult	
tableIndexClustered	
tableIndexHashed	
tableIndexOther	
tableIndexStatistic	
typeNoNulls	
typeNullable	
typeNullableUnknown	
typePredBasic	
typePredChar	
typePredNone	
typeSearchable	
versionColumnNotPseudo	
versionColumnPseudo	
versionColumnUnknown	

Public Methods

Name	Description
allProceduresAreCallable	
allTablesAreSelectable	
dataDefinitionCausesTransactionCommit	
dataDefinitionIgnoredInTransactions	
doesMaxRowSizeIncludeBlobs	

getBestRowIdentifier	
getCatalogs	
getCatalogSeparator	
getCatalogTerm	
getColumnPrivileges	
getColumns	
getCrossReference	
getDatabaseProductName	
getDatabaseProductVersion	
getDefaultTransactionIsolation	
getDriverMajorVersion	
getDriverMinorVersion	
getDriverName	
getDriverVersion	
getExportedKeys	
getExtraNameCharacters	
getIdentifierQuoteString	
getImportedKeys	
getIndexInfo	
getMaxBinaryLiteralLength	
getMaxCatalogNameLength	
getMaxCharLiteralLength	
getMaxColumnNameLength	
getMaxColumnsInGroupBy	
getMaxColumnsInIndex	
getMaxColumnsInOrderBy	
getMaxColumnsInSelect	

getMaxColumnsInTable	
getMaxConnections	
getMaxCursorNameLength	
getMaxIndexLength	
getMaxProcedureNameLength	
getMaxRowSize	
getMaxSchemaNameLength	
getMaxStatementLength	
getMaxStatements	
getMaxTableNameLength	
getMaxTablesInSelect	
getMaxUserNameLength	
getNumericFunctions	
getPrimaryKeys	
getProcedureColumns	
getProcedures	
getProcedureTerm	
getSchemas	
getSchemaTerm	
getSearchStringEscape	
getSQLKeywords	
getStringFunctions	
getSystemFunctions	
getTablePrivileges	
getTables	
getTableTypes	
getTimeDateFunctions	

getTypeInfo	
getURL	
getUserName	
getVersionColumns	
isCatalogAtStart	
isReadOnly	
nullPlusNonNullIsNull	
nullsAreSortedAtEnd	
nullsAreSortedAtStart	
nullsAreSortedHigh	
nullsAreSortedLow	
storesLowerCaseIdentifiers	
storesLowerCaseQuotedIdentifiers	
storesMixedCaseIdentifiers	
storesMixedCaseQuotedIdentifiers	
storesUpperCaseIdentifiers	
storesUpperCaseQuotedIdentifiers	
supportsAlterTableWithAddColumn	
supportsAlterTableWithDropColumn	
supportsANSI92EntryLevelSQL	
supportsANSI92FullSQL	
supportsANSI92IntermediateSQL	
supportsCatalogsInDataManipulation	
supportsCatalogsInIndexDefinitions	
supportsCatalogsInPrivilegeDefinitions	
supportsCatalogsInProcedureCalls	
supportsCatalogsInTableDefinitions	

supportsColumnAliasing	
supportsConvert	Overloaded.
supportsCoreSQLGrammar	
supportsCorrelatedSubqueries	
supportsDataDefinitionAndDataManipulationTransactions	
supportsDataManipulationTransactionsOnly	
supportsDifferentTableCorrelationNames	
supportsExpressionsInOrderBy	
supportsExtendedSQLGrammar	
supportsFullOuterJoins	
supportsGroupBy	
supportsGroupByBeyondSelect	
supportsGroupByUnrelated	
supportsIntegrityEnhancementFacility	
supportsLikeEscapeClause	
supportsLimitedOuterJoins	
supportsMinimumSQLGrammar	
supportsMixedCaseIdentifiers	
supportsMixedCaseQuotedIdentifiers	
supportsMultipleResultSets	
supportsMultipleTransactions	
supportsNonNullableColumns	
supportsOpenCursorsAcrossCommit	
supportsOpenCursorsAcrossRollback	
supportsOpenStatementsAcrossCommit	
supportsOpenStatementsAcrossRollback	

supportsOrderByUnrelated	
supportsOuterJoins	
supportsPositionedDelete	
supportsPositionedUpdate	
supportsSchemasInDataManipulation	
supportsSchemasInIndexDefinitions	
supportsSchemasInPrivilegeDefinitions	
supportsSchemasInProcedureCalls	
supportsSchemasInTableDefinitions	
supportsSelectForUpdate	
supportsStoredProcedures	
supportsSubqueriesInComparisons	
supportsSubqueriesInExists	
supportsSubqueriesInIns	
supportsSubqueriesInQuantifieds	
supportsTableCorrelationNames	
supportsTransactionIsolationLevel	
supportsTransactions	
supportsUnion	
supportsUnionAll	
usesLocalFilePerTable	
usesLocalFiles	

See Also

Reference

DatabaseMetaData Interface

Concepts

java.sql Package

DatabaseMetaData Fields

Public Fields

Name	Description
bestRowNotPseudo	
bestRowPseudo	
bestRowSession	
bestRowTemporary	
bestRowTransaction	
bestRowUnknown	
columnNoNulls	
columnNullable	
columnNullableUnknown	
importedKeyCascade	
importedKeyInitiallyDeferred	
importedKeyInitiallyImmediate	
importedKeyNoAction	
importedKeyNotDeferrable	
importedKeyRestrict	
importedKeySetDefault	
importedKeySetNull	
procedureColumnIn	
procedureColumnInOut	
procedureColumnOut	
procedureColumnResult	
procedureColumnReturn	
procedureColumnUnknown	
procedureNoNulls	

procedureNoResult	
procedureNullable	
procedureNullableUnknown	
procedureResultUnknown	
procedureReturnsResult	
tableIndexClustered	
tableIndexHashed	
tableIndexOther	
tableIndexStatistic	
typeNoNulls	
typeNullable	
typeNullableUnknown	
typePredBasic	
typePredChar	
typePredNone	
typeSearchable	
versionColumnNotPseudo	
versionColumnPseudo	
versionColumnUnknown	

See Also

Reference

DatabaseMetaData Interface

Concepts

java.sql Package

DatabaseMetaData.bestRowNotPseudo Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int bestRowNotPseudo;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.bestRowPseudo Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int bestRowPseudo;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.bestRowSession Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int bestRowSession;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.bestRowTemporary Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int bestRowTemporary;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.bestRowTransaction Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int bestRowTransaction;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.bestRowUnknown Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int bestRowUnknown;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.columnNoNulls Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int columnNoNulls;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.columnNullable Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int columnNullable;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.columnNullableUnknown Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int columnNullableUnknown;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.importedKeyCascade Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int importedKeyCascade;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.importedKeyInitiallyDeferred Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int importedKeyInitiallyDeferred;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.importedKeyInitiallyImmediate Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int importedKeyInitiallyImmediate;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.importedKeyNoAction Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int importedKeyNoAction;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.importedKeyNotDeferrable Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int importedKeyNotDeferrable;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.importedKeyRestrict Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int importedKeyRestrict;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.importedKeySetDefault Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int importedKeySetDefault;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.importedKeySetNull Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int importedKeySetNull;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureColumnIn Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureColumnIn;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureColumnInOut Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureColumnInOut;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureColumnOut Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureColumnOut;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureColumnResult Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureColumnResult;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureColumnReturn Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureColumnReturn;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureColumnUnknown Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureColumnUnknown;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureNoNulls Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureNoNulls;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureNoResult Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureNoResult;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureNullable Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureNullable;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureNullableUnknown Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureNullableUnknown;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureResultUnknown Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureResultUnknown;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.procedureReturnsResult Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int procedureReturnsResult;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.tableIndexClustered Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final short tableIndexClustered;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.tableIndexHashed Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final short tableIndexHashed;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.tableIndexOther Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final short tableIndexOther;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.tableIndexStatistic Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final short tableIndexStatistic;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.typeNoNulls Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int typeNoNulls;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.typeNullable Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int typeNullable;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.typeNullableUnknown Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int typeNullableUnknown;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.typePredBasic Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int typePredBasic;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.typePredChar Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int typePredChar;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.typePredNone Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int typePredNone;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.typeSearchable Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int typeSearchable;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.versionColumnNotPseudo Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int versionColumnNotPseudo;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.versionColumnPseudo Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int versionColumnPseudo;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.versionColumnUnknown Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int versionColumnUnknown;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData Methods

Public Methods

Name	Description
allProceduresAreCallable	
allTablesAreSelectable	
dataDefinitionCausesTransactionCommit	
dataDefinitionIgnoredInTransactions	
doesMaxRowSizeIncludeBlobs	
getBestRowIdentifier	
getCatalogs	
getCatalogSeparator	
getCatalogTerm	
getColumnPrivileges	
getColumns	
getCrossReference	
getDatabaseProductName	
getDatabaseProductVersion	
getDefaultTransactionIsolation	
getDriverMajorVersion	
getDriverMinorVersion	
getDriverName	
getDriverVersion	
getExportedKeys	
getExtraNameCharacters	
getIdentifierQuoteString	
getImportedKeys	
getIndexInfo	

getMaxBinaryLiteralLength	
getMaxCatalogNameLength	
getMaxCharLiteralLength	
getMaxColumnNameLength	
getMaxColumnsInGroupBy	
getMaxColumnsInIndex	
getMaxColumnsInOrderBy	
getMaxColumnsInSelect	
getMaxColumnsInTable	
getMaxConnections	
getMaxCursorNameLength	
getMaxIndexLength	
getMaxProcedureNameLength	
getMaxRowSize	
getMaxSchemaNameLength	
getMaxStatementLength	
getMaxStatements	
getMaxTableNameLength	
getMaxTablesInSelect	
getMaxUserNameLength	
getNumericFunctions	
getPrimaryKeys	
getProcedureColumns	
getProcedures	
getProcedureTerm	
getSchemas	
getSchemaTerm	

getSearchStringEscape	
getSQLKeywords	
getStringFunctions	
getSystemFunctions	
getTablePrivileges	
getTables	
getTableTypes	
getTimeDateFunctions	
getTypeInfo	
getURL	
getUserName	
getVersionColumns	
isCatalogAtStart	
isReadOnly	
nullPlusNonNullIsNull	
nullsAreSortedAtEnd	
nullsAreSortedAtStart	
nullsAreSortedHigh	
nullsAreSortedLow	
storesLowerCaseIdentifiers	
storesLowerCaseQuotedIdentifiers	
storesMixedCaseIdentifiers	
storesMixedCaseQuotedIdentifiers	
storesUpperCaseIdentifiers	
storesUpperCaseQuotedIdentifiers	
supportsAlterTableWithAddColumn	
supportsAlterTableWithDropColumn	

supportsANSI92EntryLevelSQL	
supportsANSI92FullSQL	
supportsANSI92IntermediateSQL	
supportsCatalogsInDataManipulation	
supportsCatalogsInIndexDefinitions	
supportsCatalogsInPrivilegeDefinitions	
supportsCatalogsInProcedureCalls	
supportsCatalogsInTableDefinitions	
supportsColumnAliasing	
supportsConvert	Overloaded.
supportsCoreSQLGrammar	
supportsCorrelatedSubqueries	
supportsDataDefinitionAndDataManipulationTransactions	
supportsDataManipulationTransactionsOnly	
supportsDifferentTableCorrelationNames	
supportsExpressionsInOrderBy	
supportsExtendedSQLGrammar	
supportsFullOuterJoins	
supportsGroupBy	
supportsGroupByBeyondSelect	
supportsGroupByUnrelated	
supportsIntegrityEnhancementFacility	
supportsLikeEscapeClause	
supportsLimitedOuterJoins	
supportsMinimumSQLGrammar	
supportsMixedCaseIdentifiers	
supportsMixedCaseQuotedIdentifiers	

supportsMultipleResultSets	
supportsMultipleTransactions	
supportsNonNullableColumns	
supportsOpenCursorsAcrossCommit	
supportsOpenCursorsAcrossRollback	
supportsOpenStatementsAcrossCommit	
supportsOpenStatementsAcrossRollback	
supportsOrderByUnrelated	
supportsOuterJoins	
supportsPositionedDelete	
supportsPositionedUpdate	
supportsSchemasInDataManipulation	
supportsSchemasInIndexDefinitions	
supportsSchemasInPrivilegeDefinitions	
supportsSchemasInProcedureCalls	
supportsSchemasInTableDefinitions	
supportsSelectForUpdate	
supportsStoredProcedures	
supportsSubqueriesInComparisons	
supportsSubqueriesInExists	
supportsSubqueriesInIns	
supportsSubqueriesInQuantifieds	
supportsTableCorrelationNames	
supportsTransactionIsolationLevel	
supportsTransactions	
supportsUnion	
supportsUnionAll	

usesLocalFilePerTable	
usesLocalFiles	

See Also

Reference

DatabaseMetaData Interface

Concepts

java.sql Package

DatabaseMetaData.allProceduresAreCallable Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean allProceduresAreCallable() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.allTablesAreSelectable Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean allTablesAreSelectable() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.dataDefinitionCausesTransactionCommit Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean dataDefinitionCausesTransactionCommit() throws java.sql.SQLExceptio
n;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.dataDefinitionIgnoredInTransactions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean dataDefinitionIgnoredInTransactions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.doesMaxRowSizeIncludeBlobs Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean doesMaxRowSizeIncludeBlobs() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getBestRowIdentifier Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getBestRowIdentifier(
    java.lang.String catalog,
    java.lang.String schema,
    java.lang.String table,
    int scope,
    boolean nullable) throws java.sql.SQLException;
```

Parameters

catalog

schema

table

scope

nullable

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getCatalogs Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getCatalogs() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getCatalogSeparator Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getCatalogSeparator() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getCatalogTerm Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getCatalogTerm() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getColumnPrivileges Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getColumnPrivileges(
    java.lang.String catalog,
    java.lang.String schema,
    java.lang.String table,
    java.lang.String colNamePattern) throws java.sql.SQLException;
```

Parameters

catalog

schema

table

colNamePattern

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getColumns Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getColumns(
    java.lang.String catalog,
    java.lang.String schemaPattern,
    java.lang.String tableNamePattern,
    java.lang.String columnNamePattern) throws java.sql.SQLException;
```

Parameters

catalog

schemaPattern

tableNamePattern

columnNamePattern

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getCrossReference Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getCrossReference(
    java.lang.String primaryCatalog,
    java.lang.String primarySchema,
    java.lang.String primaryTable,
    java.lang.String foreignCatalog,
    java.lang.String foreignSchema,
    java.lang.String foreignTable) throws java.sql.SQLException;
```

Parameters

primaryCatalog

primarySchema

primaryTable

foreignCatalog

foreignSchema

foreignTable

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getDatabaseProductName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getDatabaseProductName() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getDatabaseProductVersion Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getDatabaseProductVersion() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getDefaultTransactionIsolation Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getDefaultTransactionIsolation() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getDriverMajorVersion Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getDriverMajorVersion();
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getDriverMinorVersion Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getDriverMinorVersion();
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getDriverName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getDriverName() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getDriverVersion Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getDriverVersion() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getExportedKeys Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getExportedKeys(
    java.lang.String catalog,
    java.lang.String schema,
    java.lang.String table) throws java.sql.SQLException;
```

Parameters

catalog

schema

table

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getExtraNameCharacters Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getExtraNameCharacters() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getIdentifierQuoteString Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getIdentifierQuoteString() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getImportedKeys Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getImportedKeys(
    java.lang.String catalog,
    java.lang.String schema,
    java.lang.String table) throws java.sql.SQLException;
```

Parameters

catalog

schema

table

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getIndexInfo Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getIndexInfo(
    java.lang.String catalog,
    java.lang.String schema,
    java.lang.String table,
    boolean unique,
    boolean approximate) throws java.sql.SQLException;
```

Parameters

catalog

schema

table

unique

approximate

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxBinaryLiteralLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxBinaryLiteralLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxCatalogNameLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxCatalogNameLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxCharLiteralLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxCharLiteralLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxColumnNameLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxColumnNameLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxColumnsInGroupBy Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxColumnsInGroupBy() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxColumnsInIndex Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxColumnsInIndex() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxColumnsInOrderBy Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxColumnsInOrderBy() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxColumnsInSelect Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxColumnsInSelect() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxColumnsInTable Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxColumnsInTable() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxConnections Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxConnections() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxCursorNameLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxCursorNameLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxIndexLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxIndexLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxProcedureNameLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxProcedureNameLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxRowSize Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxRowSize() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxSchemaNameLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxSchemaNameLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxStatementLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxStatementLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxStatements Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxStatements() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxTableNameLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxTableNameLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getMaxTablesInSelect Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxTablesInSelect() throws java.sql.SQLException;
```

See Also
Reference
DatabaseMetaData Interface
Concepts
DatabaseMetaData Members
java.sql Package

DatabaseMetaData.getMaxUserNameLength Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxUserNameLength() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getNumericFunctions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getNumericFunctions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getPrimaryKeys Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getPrimaryKeys(
    java.lang.String catalog,
    java.lang.String schema,
    java.lang.String table) throws java.sql.SQLException;
```

Parameters

catalog

schema

table

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getProcedureColumns Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getProcedureColumns(
    java.lang.String catalog,
    java.lang.String schemaPattern,
    java.lang.String procedureNamePattern,
    java.lang.String columnNamePattern) throws java.sql.SQLException;
```

Parameters

catalog

schemaPattern

procedureNamePattern

columnNamePattern

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getProcedures Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getProcedures(
    java.lang.String catalog,
    java.lang.String schemaPattern,
    java.lang.String procedureNamePattern) throws java.sql.SQLException;
```

Parameters

catalog

schemaPattern

procedureNamePattern

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getProcedureTerm Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getProcedureTerm() throws java.sql.SQLException;
```

See Also
Reference
DatabaseMetaData Interface
Concepts
DatabaseMetaData Members
java.sql Package

DatabaseMetaData.getSchemas Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getSchemas() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getSchemaTerm Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getSchemaTerm() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getSearchStringEscape Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getSearchStringEscape() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getSQLKeywords Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getSQLKeywords() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getStringFunctions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getStringFunctions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getSystemFunctions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getSystemFunctions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getTablePrivileges Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getTablePrivileges(
    java.lang.String catalog,
    java.lang.String schemaPattern,
    java.lang.String tableNamePattern) throws java.sql.SQLException;
```

Parameters

catalog

schemaPattern

tableNamePattern

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getTables Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getTables(
    java.lang.String catalog,
    java.lang.String schemaPattern,
    java.lang.String tableNamePattern,
    java.lang.String[] types) throws java.sql.SQLException;
```

Parameters

catalog

schemaPattern

tableNamePattern

types

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getTableTypes Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getTableTypes() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getTimeDateFunctions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getTimeDateFunctions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getTypeInfo Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getTypeInfo() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getURL Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getURL() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getUserName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getUserName() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.getVersionColumns Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getVersionColumns(
    java.lang.String catalog,
    java.lang.String schema,
    java.lang.String table) throws java.sql.SQLException;
```

Parameters

catalog

schema

table

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.isCatalogAtStart Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isCatalogAtStart() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.isReadOnly Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isReadOnly() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.nullPlusNonNullIsNull Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean nullPlusNonNullIsNull() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.nullsAreSortedAtEnd Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean nullsAreSortedAtEnd() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.nullsAreSortedAtStart Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean nullsAreSortedAtStart() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.nullsAreSortedHigh Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean nullsAreSortedHigh() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.nullsAreSortedLow Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean nullsAreSortedLow() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.storesLowerCaseIdentifiers Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean storesLowerCaseIdentifiers() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.storesLowerCaseQuotedIdentifiers Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean storesLowerCaseQuotedIdentifiers() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.storesMixedCaseIdentifiers Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean storesMixedCaseIdentifiers() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.storesMixedCaseQuotedIdentifiers Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean storesMixedCaseQuotedIdentifiers() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.storesUpperCaseIdentifiers Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean storesUpperCaseIdentifiers() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.storesUpperCaseQuotedIdentifiers Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean storesUpperCaseQuotedIdentifiers() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsAlterTableWithAddColumn Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsAlterTableWithAddColumn() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsAlterTableWithDropColumn Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsAlterTableWithDropColumn() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsANSI92EntryLevelSQL Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsANSI92EntryLevelSQL() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsANSI92FullSQL Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsANSI92FullSQL() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsANSI92IntermediateSQL Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsANSI92IntermediateSQL() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsCatalogsInDataManipulation Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsCatalogsInDataManipulation() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsCatalogsInIndexDefinitions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsCatalogsInIndexDefinitions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsCatalogsInPrivilegeDefinitions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsCatalogsInPrivilegeDefinitions() throws java.sql.SQLExcepti
on;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsCatalogsInProcedureCalls Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsCatalogsInProcedureCalls() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsCatalogsInTableDefinitions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsCatalogsInTableDefinitions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsColumnAliasing Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsColumnAliasing() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsConvert Method

Overload List

Name	Description
DatabaseMetaData.supportsConvert ()	
DatabaseMetaData.supportsConvert (int, int)	

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsConvert Method ()

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsConvert() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsConvert Method (Int32, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsConvert(
    int fromType,
    int toType) throws java.sql.SQLException;
```

Parameters

fromType

toType

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsCoreSQLGrammar Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsCoreSQLGrammar() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsCorrelatedSubqueries Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsCorrelatedSubqueries() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsDataDefinitionAndDataManipulationTransactions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsDataDefinitionAndDataManipulationTransactions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsDataManipulationTransactionsOnly Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsDataManipulationTransactionsOnly() throws java.sql.SQLExcep
tion;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsDifferentTableCorrelationNames Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsDifferentTableCorrelationNames() throws java.sql.SQLExcepti
on;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsExpressionsInOrderBy Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsExpressionsInOrderBy() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsExtendedSQLGrammar Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsExtendedSQLGrammar() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsFullOuterJoins Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsFullOuterJoins() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsGroupBy Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsGroupBy() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsGroupByBeyondSelect Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsGroupByBeyondSelect() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsGroupByUnrelated Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsGroupByUnrelated() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsIntegrityEnhancementFacility Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsIntegrityEnhancementFacility() throws java.sql.SQLException
;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsLikeEscapeClause Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsLikeEscapeClause() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsLimitedOuterJoins Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsLimitedOuterJoins() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsMinimumSQLGrammar Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsMinimumSQLGrammar() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsMixedCaseIdentifiers Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsMixedCaseIdentifiers() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsMixedCaseQuotedIdentifiers Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsMixedCaseQuotedIdentifiers() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsMultipleResultSets Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsMultipleResultSets() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsMultipleTransactions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsMultipleTransactions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsNonNullableColumns Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsNonNullableColumns() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsOpenCursorsAcrossCommit Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsOpenCursorsAcrossCommit() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsOpenCursorsAcrossRollback Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsOpenCursorsAcrossRollback() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsOpenStatementsAcrossRollback Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsOpenStatementsAcrossRollback() throws java.sql.SQLException
;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsOrderByUnrelated Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsOrderByUnrelated() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsOuterJoins Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsOuterJoins() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsPositionedDelete Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsPositionedDelete() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsPositionedUpdate Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsPositionedUpdate() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSchemasInDataManipulation Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSchemasInDataManipulation() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSchemasInIndexDefinitions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSchemasInIndexDefinitions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSchemasInPrivilegeDefinitions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSchemasInPrivilegeDefinitions() throws java.sql.SQLExceptio
n;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSchemasInProcedureCalls Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSchemasInProcedureCalls() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSchemasInTableDefinitions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSchemasInTableDefinitions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSelectForUpdate Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSelectForUpdate() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsStoredProcedures Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsStoredProcedures() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSubqueriesInComparisons Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSubqueriesInComparisons() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSubqueriesInExists Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSubqueriesInExists() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSubqueriesInIns Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSubqueriesInIns() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsSubqueriesInQuantifieds Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsSubqueriesInQuantifieds() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsTableCorrelationNames Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsTableCorrelationNames() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsTransactionIsolationLevel Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsTransactionIsolationLevel(
    int level) throws java.sql.SQLException;
```

Parameters

level

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsTransactions Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsTransactions() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsUnion Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsUnion() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.supportsUnionAll Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean supportsUnionAll() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.usesLocalFilePerTable Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean usesLocalFilePerTable() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DatabaseMetaData.usesLocalFiles Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean usesLocalFiles() throws java.sql.SQLException;
```

See Also

Reference

DatabaseMetaData Interface

Concepts

DatabaseMetaData Members

java.sql Package

DataTruncation Class

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.DataTruncation
    extends java.sql.SQLWarning
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.sql.SQLException

java.sql.SQLWarning

java.sql.DataTruncation

See Also

Concepts

DataTruncation Members

java.sql Package

DataTruncation Members

The following tables list the members exposed by the DataTruncation type.

Public Constructors

Name	Description
DataTruncation	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
getDataSize	
getErrorCode	(inherited from SQLException)
hashCode	(inherited from Object)
getIndex	
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
getNextException	(inherited from SQLException)
getNextWarning	(inherited from SQLWarning)

GetObjectData	Overridden.
getParameter	
getRead	
getSQLState	(inherited from SQLException)
getTransferSize	
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
setNextException	(inherited from SQLException)
setNextWarning	(inherited from SQLWarning)
toString	(inherited from SQLException)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DataTruncation Class

Concepts

java.sql Package

DataTruncation Constructor

Overload List

Name	Description
DataTruncation (SerializationInfo, StreamingContext)	
DataTruncation (String, Exception)	
DataTruncation (int, boolean, boolean, int, int)	
DataTruncation (int, boolean, boolean, int, int, Exception)	

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation Constructor (SerializationInfo, StreamingContext)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
protected java.sql.DataTruncation(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation Constructor (String, Exception)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.DataTruncation(
    java.lang.String msg,
    System.Exception inner);
```

Parameters

msg

inner

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation Constructor (Int32, Boolean, Boolean, Int32, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.DataTruncation(
    int index,
    boolean parameter,
    boolean read,
    int dataSize,
    int transferSize);
```

Parameters

index

parameter

read

dataSize

transferSize

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation Constructor (Int32, Boolean, Boolean, Int32, Int32, Exception)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.DataTruncation(
    int index,
    boolean parameter,
    boolean read,
    int dataSize,
    int transferSize,
    System.Exception inner);
```

Parameters

index

parameter

read

dataSize

transferSize

inner

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
getDataSize	
getErrorCode	(inherited from SQLException)
hashCode	(inherited from Object)
getIndex	
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
getNextException	(inherited from SQLException)
getNextWarning	(inherited from SQLWarning)
GetObjectData	Overridden.
getParameter	
getRead	
getSQLState	(inherited from SQLException)
getTransferSize	
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
setNextException	(inherited from SQLException)
setNextWarning	(inherited from SQLWarning)
toString	(inherited from SQLException)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DataTruncation Class

Concepts

java.sql Package

DataTruncation.getDataSize Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getDataSize();
```

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation.getIndex Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getIndex();
```

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation.GetObjectData Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation.getParameter Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public boolean getParameter();
```

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation.getRead Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public boolean getRead();
```

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation.getTransferSize Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getTransferSize();
```

See Also

Reference

DataTruncation Class

Concepts

DataTruncation Members

java.sql Package

DataTruncation Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

DataTruncation Class

Concepts

java.sql Package

Date Class

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.Date
    extends java.util.Date
```

Inheritance Hierarchy

java.lang.Object

  java.util.Date

   java.sql.Date

See Also

Concepts

Date Members

java.sql Package

Date Members

The following tables list the members exposed by the Date type.

Public Constructors

Name	Description
Date	Overloaded.

Public Methods

Name	Description
after	Determines whether the Date object is after a given date. (inherited from Date)
before	Determines whether the Date object is before a given date. (inherited from Date)
compareTo	Compares two dates. (inherited from Date)
equals	Determines whether two Date objects are equal. (inherited from Date)
getDate	Gets a value indicating the date. (inherited from Date)
getDay	Gets a value indicating the day of the week. (inherited from Date)
hashCode	Generates a hash of the date. (inherited from Date)
getHours	Overridden.
getMinutes	Overridden.
getMonth	Gets a value indicating the month. (inherited from Date)
getSeconds	Overridden.
getTime	Gets a value indicating the time in milliseconds since epoch. (inherited from Date)
getTimezoneOffset	Gets a value indicating the time zone offset in milliseconds. (inherited from Date)
getClass	(inherited from Object)
getYear	Gets a value indicating the year. (inherited from Date)
clone	(inherited from Date)
setDate	Sets a value indicating the date. (inherited from Date)
setHours	Overridden.
setMinutes	Overridden.
setMonth	Sets a value indicating the month. (inherited from Date)

setSeconds	Overridden.
setTime	Sets the time to the value represented in milliseconds since epoch. (inherited from Date)
setYear	Sets a value indicating the year. (inherited from Date)
ToBoolean	Converts an instance of a Date object to a Boolean. (inherited from Date)
ToByte	Converts an instance of a Date object to a Byte. (inherited from Date)
ToChar	Converts an instance of a Date object to a Char. (inherited from Date)
ToDateTime	Converts an instance of a Date object to a DateTime. (inherited from Date)
ToDecimal	Converts an instance of a Date object to a Decimal. (inherited from Date)
ToDouble	Converts an instance of a Date object to a Double. (inherited from Date)
toGMTString	Displays a value representing the date/time of a Date object in GMT. (inherited from Date)
ToInt16	Converts an instance of a Date object to an Int16. (inherited from Date)
ToInt32	Converts an instance of a Date object to an Int32. (inherited from Date)
ToInt64	Converts an instance of a Date object to an Int64. (inherited from Date)
toLocaleString	Displays a value representing the date/time of a Date object displayed in the format of current locale. (inherited from Date)
ToSByte	Converts an instance of a Date object to a SByte. (inherited from Date)
ToSingle	Converts an instance of a Date object to a Single. (inherited from Date)
toString	Overloaded. Overridden.
ToType	Converts an instance of a Date object to a given Type. (inherited from Date)
ToUInt16	Converts an instance of a Date object to an UInt16. (inherited from Date)
ToUInt32	Converts an instance of a Date object to an UInt32. (inherited from Date)
ToUInt64	Converts an instance of a Date object to an UInt64. (inherited from Date)
valueOf	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Date Class

Concepts

java.sql Package

Date Constructor

Overload List

Name	Description
Date (long)	
Date (int, int, int)	

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date Constructor (Int64)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.Date(
    long msSinceEpoch);
```

Parameters
msSinceEpoch

See Also
Reference
Date Class
Concepts
Date Members
java.sql Package

Date Constructor (Int32, Int32, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.Date(
    int year,
    int month,
    int date);
```

Parameters

year

month

date

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date Methods

Public Methods

Name	Description
after	Determines whether the Date object is after a given date. (inherited from Date)
before	Determines whether the Date object is before a given date. (inherited from Date)
compareTo	Compares two dates. (inherited from Date)
equals	Determines whether two Date objects are equal. (inherited from Date)
getDate	Gets a value indicating the date. (inherited from Date)
getDay	Gets a value indicating the day of the week. (inherited from Date)
hashCode	Generates a hash of the date. (inherited from Date)
getHours	Overridden.
getMinutes	Overridden.
getMonth	Gets a value indicating the month. (inherited from Date)
getSeconds	Overridden.
getTime	Gets a value indicating the time in milliseconds since epoch. (inherited from Date)
getTimezoneOffset	Gets a value indicating the time zone offset in milliseconds. (inherited from Date)
getClass	(inherited from Object)
getYear	Gets a value indicating the year. (inherited from Date)
clone	(inherited from Date)
setDate	Sets a value indicating the date. (inherited from Date)
setHours	Overridden.
setMinutes	Overridden.
setMonth	Sets a value indicating the month. (inherited from Date)
setSeconds	Overridden.
setTime	Sets the time to the value represented in milliseconds since epoch. (inherited from Date)
setYear	Sets a value indicating the year. (inherited from Date)

ToBoolean	Converts an instance of a Date object to a Boolean. (inherited from Date)
ToByte	Converts an instance of a Date object to a Byte. (inherited from Date)
ToChar	Converts an instance of a Date object to a Char. (inherited from Date)
ToDateTime	Converts an instance of a Date object to a DateTime. (inherited from Date)
ToDecimal	Converts an instance of a Date object to a Decimal. (inherited from Date)
ToDouble	Converts an instance of a Date object to a Double. (inherited from Date)
toGMTString	Displays a value representing the date/time of a Date object in GMT. (inherited from Date)
ToInt16	Converts an instance of a Date object to an Int16. (inherited from Date)
ToInt32	Converts an instance of a Date object to an Int32. (inherited from Date)
ToInt64	Converts an instance of a Date object to an Int64. (inherited from Date)
toLocaleString	Displays a value representing the date/time of a Date object displayed in the format of current locale. (inherited from Date)
ToSByte	Converts an instance of a Date object to a SByte. (inherited from Date)
ToSingle	Converts an instance of a Date object to a Single. (inherited from Date)
toString	Overloaded. Overridden.
ToType	Converts an instance of a Date object to a given Type. (inherited from Date)
ToUInt16	Converts an instance of a Date object to an UInt16. (inherited from Date)
ToUInt32	Converts an instance of a Date object to an UInt32. (inherited from Date)
ToUInt64	Converts an instance of a Date object to an UInt64. (inherited from Date)
valueOf	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Date Class

Concepts

java.sql Package

Date.getHours Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getHours();
```

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.getMinutes Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getMinutes();
```

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.getSeconds Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getSeconds();
```

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.setHours Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setHours(
    int hour);
```

Parameters

hour

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.setMinutes Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setMinutes(
    int minute);
```

Parameters

minute

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.setSeconds Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setSeconds(
    int second);
```

Parameters

second

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.toString Method

Overload List

Name	Description
Date.ToString ()	
Date.ToString (IFormatProvider)	Converts an instance of a Date object to a String.

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.toString Method ()

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.valueOf Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static java.sql.Date valueOf(
    java.lang.String strDate);
```

Parameters

strDate

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Driver Interface

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public interface java.sql.Driver
```

See Also

Concepts

Driver Members

java.sql Package

Driver Members

The following tables list the members exposed by the Driver type.

Public Methods

Name	Description
acceptsURL	
connect	
getMajorVersion	
getMinorVersion	
getPropertyInfo	
jdbcCompliant	

See Also

Reference

Driver Interface

Concepts

java.sql Package

Driver Methods

Public Methods

Name	Description
acceptsURL	
connect	
getMajorVersion	
getMinorVersion	
getPropertyInfo	
jdbcCompliant	

See Also

Reference

Driver Interface

Concepts

java.sql Package

Driver.acceptsURL Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean acceptsURL(
    java.lang.String url) throws java.sql.SQLException;
```

Parameters

url

See Also

Reference

Driver Interface

Concepts

Driver Members

java.sql Package

Driver.connect Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Connection connect(
    java.lang.String url,
    java.util.Properties info) throws java.sql.SQLException;
```

Parameters

url

info

See Also

Reference

Driver Interface

Concepts

Driver Members

java.sql Package

Driver.getMajorVersion Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMajorVersion();
```

See Also

Reference

Driver Interface

Concepts

Driver Members

java.sql Package

Driver.getMinorVersion Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMinorVersion();
```

See Also

Reference

Driver Interface

Concepts

Driver Members

java.sql Package

Driver.getPropertyInfo Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.DriverPropertyInfo[] getPropertyInfo(
    java.lang.String url,
    java.util.Properties info) throws java.sql.SQLException;
```

Parameters

url

info

See Also

Reference

Driver Interface

Concepts

Driver Members

java.sql Package

Driver.jdbcCompliant Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean jdbcCompliant();
```

See Also

Reference

Driver Interface

Concepts

Driver Members

java.sql Package

DriverManager Class

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.DriverManager
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.sql.DriverManager

See Also

Concepts

DriverManager Members

java.sql Package

DriverManager Members

The following tables list the members exposed by the DriverManager type.

Public Methods

Name	Description
deregisterDriver	
equals	(inherited from Object)
getConnection	Overloaded.
getDriver	
getDrivers	
hashCode	(inherited from Object)
getLoginTimeout	
getLogStream	
getClass	(inherited from Object)
println	
registerDriver	
setLoginTimeout	
setLogStream	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

DriverManager Class

Concepts

java.sql Package

DriverManager Methods

Public Methods

Name	Description
deregisterDriver	
equals	(inherited from Object)
getConnection	Overloaded.
getDriver	
getDrivers	
hashCode	(inherited from Object)
getLoginTimeout	
getLogStream	
getClass	(inherited from Object)
println	
registerDriver	
setLoginTimeout	
setLogStream	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

DriverManager Class

Concepts

java.sql Package

DriverManager.deregisterDriver Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static void deregisterDriver(
    java.sql.Driver driver) throws java.sql.SQLException;
```

Parameters

driver

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.getConnection Method

Overload List

Name	Description
DriverManager.getConnection (String)	
DriverManager.getConnection (String, Properties)	
DriverManager.getConnection (String, String, String)	

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.getConnection Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static java.sql.Connection getConnection(
    java.lang.String url) throws java.sql.SQLException;
```

Parameters

url

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.getConnection Method (String, Properties)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static java.sql.Connection getConnection(
    java.lang.String url,
    java.util.Properties info) throws java.sql.SQLException;
```

Parameters

url

info

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.getConnection Method (String, String, String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static java.sql.Connection getConnection(
    java.lang.String url,
    java.lang.String user,
    java.lang.String password) throws java.sql.SQLException;
```

Parameters

url

user

password

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.getDriver Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static java.sql.Driver getDriver(
    java.lang.String url) throws java.sql.SQLException;
```

Parameters

url

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.getDrivers Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Enumeration getDrivers();
```

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.getLoginTimeout Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static int getLoginTimeout();
```

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.getLogStream Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static java.io.PrintStream getLogStream();
```

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.println Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static void println(
    java.lang.String message);
```

Parameters

message

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.registerDriver Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static void registerDriver(
    java.sql.Driver driver) throws java.sql.SQLException;
```

Parameters

driver

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.setLoginTimeout Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static void setLoginTimeout(
    int seconds);
```

Parameters

seconds

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverManager.setLogStream Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static void setLogStream(
    java.io.PrintStream out);
```

Parameters

out

See Also

Reference

DriverManager Class

Concepts

DriverManager Members

java.sql Package

DriverPropertyInfo Class

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.DriverPropertyInfo
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.sql.DriverPropertyInfo

See Also

Concepts

DriverPropertyInfo Members

java.sql Package

DriverPropertyInfo Members

The following tables list the members exposed by the DriverPropertyInfo type.

Public Constructors

Name	Description
DriverPropertyInfo	

Public Fields

Name	Description
choices	
description	
name	
required	
value	

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

DriverPropertyInfo Class

Concepts

java.sql Package

DriverPropertyInfo Fields

Public Fields

Name	Description
choices	
description	
name	
required	
value	

See Also

Reference

DriverPropertyInfo Class

Concepts

java.sql Package

DriverPropertyInfo.choices Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String[] choices;
```

See Also

Reference

DriverPropertyInfo Class

Concepts

DriverPropertyInfo Members

java.sql Package

DriverPropertyInfo.description Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String description;
```

See Also

Reference

DriverPropertyInfo Class

Concepts

DriverPropertyInfo Members

java.sql Package

DriverPropertyInfo.name Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String name;
```

See Also

Reference

DriverPropertyInfo Class

Concepts

DriverPropertyInfo Members

java.sql Package

DriverPropertyInfo.required Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public boolean required;
```

See Also

Reference

DriverPropertyInfo Class

Concepts

DriverPropertyInfo Members

java.sql Package

DriverPropertyInfo.value Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String value;
```

See Also

Reference

DriverPropertyInfo Class

Concepts

DriverPropertyInfo Members

java.sql Package

DriverPropertyInfo Constructor

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.DriverPropertyInfo(
    java.lang.String name,
    java.lang.String value);
```

Parameters

name

value

See Also

Reference

DriverPropertyInfo Class

Concepts

DriverPropertyInfo Members

java.sql Package

DriverPropertyInfo Methods

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

DriverPropertyInfo Class

Concepts

java.sql Package

PreparedStatement Interface

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public interface java.sql.PreparedStatement
    extends java.sql.Statement
```

See Also

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement Members

The following tables list the members exposed by the PreparedStatement type.

Public Methods

Name	Description
clearParameters	
execute	
executeQuery	
executeUpdate	
setAsciiStream	
setBigDecimal	
setBinaryStream	
setBoolean	
setByte	
setBytes	
setDate	
setDouble	
setFloat	
setInt	
setLong	
setNull	
setObject	Overloaded.
setShort	
setString	
setTime	
setTimestamp	
setUnicodeStream	

See Also

Reference

PreparedStatement Interface

Concepts
java.sql Package

PreparedStatement Methods

Public Methods

Name	Description
clearParameters	
execute	
executeQuery	
executeUpdate	
setAsciiStream	
setBigDecimal	
setBinaryStream	
setBoolean	
setByte	
setBytes	
setDate	
setDouble	
setFloat	
setInt	
setLong	
setNull	
setObject	Overloaded.
setShort	
setString	
setTime	
setTimestamp	
setUnicodeStream	

See Also

Reference

PreparedStatement Interface

Concepts

java.sql Package

PreparedStatement.clearParameters Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void clearParameters() throws java.sql.SQLException;
```

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.execute Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean execute() throws java.sql.SQLException;
```

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.executeQuery Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet executeQuery() throws java.sql.SQLException;
```

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.executeUpdate Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int executeUpdate() throws java.sql.SQLException;
```

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setAsciiStream Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setAsciiStream(
    int parameterIndex,
    java.io.InputStream x,
    int length) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

length

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setBigDecimal Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setBigDecimal(
    int parameterIndex,
    java.math.BigDecimal x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setBinaryStream Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setBinaryStream(
    int parameterIndex,
    java.io.InputStream x,
    int length) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

length

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setBoolean Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setBoolean(
    int parameterIndex,
    boolean x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setByte Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setByte(
    int parameterIndex,
    byte x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setBytes Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setBytes(
    int parameterIndex,
    byte[] x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setDate Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setDate(
    int parameterIndex,
    java.sql.Date x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also
Reference
PreparedStatement Interface
Concepts
PreparedStatement Members
java.sql Package

PreparedStatement.setDouble Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setDouble(
    int parameterIndex,
    double x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setFloat Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setFloat(
    int parameterIndex,
    float x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also
Reference
PreparedStatement Interface
Concepts
PreparedStatement Members
java.sql Package

PreparedStatement.setInt Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setInt(
    int parameterIndex,
    int x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setLong Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setLong(
    int parameterIndex,
    long x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setNull Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setNull(
    int parameterIndex,
    int sqlType) throws java.sql.SQLException;
```

Parameters

parameterIndex

sqlType

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setObject Method

Overload List

Name	Description
PreparedStatement.setObject (int, Object)	
PreparedStatement.setObject (int, Object, int)	
PreparedStatement.setObject (int, Object, int, int)	

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setObject Method (Int32, Object)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setObject(
    int parameterIndex,
    java.lang.Object x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also
Reference
PreparedStatement Interface
Concepts
PreparedStatement Members
java.sql Package

PreparedStatement.setObject Method (Int32, Object, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setObject(
    int parameterIndex,
    java.lang.Object x,
    int targetSqlType) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

targetSqlType

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setObject Method (Int32, Object, Int32, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setObject(
    int parameterIndex,
    java.lang.Object x,
    int targetSqlType,
    int scale) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

targetSqlType

scale

See Also
Reference
PreparedStatement Interface
Concepts
PreparedStatement Members
java.sql Package

PreparedStatement.setShort Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setShort(
    int parameterIndex,
    short x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setString Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setString(
    int parameterIndex,
    java.lang.String x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also
Reference
PreparedStatement Interface
Concepts
PreparedStatement Members
java.sql Package

PreparedStatement.setTime Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setTime(
    int parameterIndex,
    java.sql.Time x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setTimestamp Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setTimestamp(
    int parameterIndex,
    java.sql.Timestamp x) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

PreparedStatement.setUnicodeStream Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setUnicodeStream(
    int parameterIndex,
    java.io.InputStream x,
    int length) throws java.sql.SQLException;
```

Parameters

parameterIndex

x

length

See Also

Reference

PreparedStatement Interface

Concepts

PreparedStatement Members

java.sql Package

ResultSet Interface

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public interface java.sql.ResultSet
```

See Also

Concepts

ResultSet Members

java.sql Package

ResultSet Members

The following tables list the members exposed by the ResultSet type.

Public Methods

Name	Description
clearWarnings	
close	
findColumn	
getAsciiStream	Overloaded.
getBigDecimal	Overloaded.
getBinaryStream	Overloaded.
getBoolean	Overloaded.
getByte	Overloaded.
getBytes	Overloaded.
getCursorName	
getDate	Overloaded.
getDouble	Overloaded.
getFloat	Overloaded.
getInt	Overloaded.
getLong	Overloaded.
getMetaData	
getObject	Overloaded.
getShort	Overloaded.
getString	Overloaded.
getTime	Overloaded.
getTimestamp	Overloaded.
getUnicodeStream	Overloaded.
getWarnings	

next	
wasNull	

See Also

Reference

ResultSet Interface

Concepts

java.sql Package

ResultSet Methods

Public Methods

Name	Description
clearWarnings	
close	
findColumn	
getAsciiStream	Overloaded.
getBigDecimal	Overloaded.
getBinaryStream	Overloaded.
getBoolean	Overloaded.
getByte	Overloaded.
getBytes	Overloaded.
getCursorName	
getDate	Overloaded.
getDouble	Overloaded.
getFloat	Overloaded.
getInt	Overloaded.
getLong	Overloaded.
getMetaData	
getObject	Overloaded.
getShort	Overloaded.
getString	Overloaded.
getTime	Overloaded.
getTimestamp	Overloaded.
getUnicodeStream	Overloaded.
getWarnings	
next	

wasNull	

See Also

Reference

ResultSet Interface

Concepts

java.sql Package

ResultSet.clearWarnings Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void clearWarnings() throws java.sql.SQLException;
```

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.close Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void close() throws java.sql.SQLException;
```

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.findColumn Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int findColumn(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getAsciiStream Method

Overload List

Name	Description
ResultSet.getAsciiStream (int)	
ResultSet.getAsciiStream (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getAsciiStream Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.io.InputStream getAsciiStream(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getAsciiStream Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.io.InputStream getAsciiStream(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBigDecimal Method

Overload List

Name	Description
ResultSet.getBigDecimal (int, int)	
ResultSet.getBigDecimal (String, int)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBigDecimal Method (Int32, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.math.BigDecimal getBigDecimal(
    int columnIndex,
    int scale) throws java.sql.SQLException;
```

Parameters

columnIndex

scale

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBigDecimal Method (String, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.math.BigDecimal getBigDecimal(
    java.lang.String columnName,
    int scale) throws java.sql.SQLException;
```

Parameters

columnName

scale

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBinaryStream Method

Overload List

Name	Description
ResultSet.getBinaryStream (int)	
ResultSet.getBinaryStream (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBinaryStream Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.io.InputStream getBinaryStream(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBinaryStream Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.io.InputStream getBinaryStream(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also
Reference
ResultSet Interface
Concepts
ResultSet Members
java.sql Package

ResultSet.getBoolean Method

Overload List

Name	Description
ResultSet.getBoolean (int)	
ResultSet.getBoolean (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBoolean Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean getBoolean(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBoolean Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean getBoolean(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getByte Method

Overload List

Name	Description
ResultSet.getByte (int)	
ResultSet.getByte (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getByte Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract byte getByte(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getByte Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract byte getByte(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBytes Method

Overload List

Name	Description
ResultSet.getBytes (int)	
ResultSet.getBytes (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBytes Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract byte[] getBytes(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getBytes Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract byte[] getBytes(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getCursorName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getCursorName() throws java.sql.SQLException;
```

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getDate Method

Overload List

Name	Description
ResultSet.getDate (int)	
ResultSet.getDate (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getDate Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Date getDate(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getDate Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Date getDate(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getDouble Method

Overload List

Name	Description
ResultSet.getDouble (int)	
ResultSet.getDouble (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getDouble Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract double getDouble(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also
Reference
ResultSet Interface
Concepts
ResultSet Members
java.sql Package

ResultSet.getDouble Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract double getDouble(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getFloat Method

Overload List

Name	Description
ResultSet.getFloat (int)	
ResultSet.getFloat (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getFloat Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract float getFloat(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getFloat Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract float getFloat(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getInt Method

Overload List

Name	Description
ResultSet.getInt (int)	
ResultSet.getInt (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getInt Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getInt(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getInt Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getInt(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getLong Method

Overload List

Name	Description
ResultSet.getLong (int)	
ResultSet.getLong (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getLong Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract long getLong(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getLong Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract long getLong(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getMetaData Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSetMetaData getMetaData() throws java.sql.SQLException;
```

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getObject Method

Overload List

Name	Description
ResultSet.getObject (int)	
ResultSet.getObject (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getObject Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object getObject(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getObject Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object getObject(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getShort Method

Overload List

Name	Description
ResultSet.getShort (int)	
ResultSet.getShort (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getShort Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract short getShort(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getShort Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract short getShort(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getString Method

Overload List

Name	Description
ResultSet.getString (int)	
ResultSet.getString (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getString Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getString(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getString Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getString(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getTime Method

Overload List

Name	Description
ResultSet.getTime (int)	
ResultSet.getTime (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getTime Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Time getTime(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getTime Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Time getTime(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getTimestamp Method

Overload List

Name	Description
ResultSet.getTimestamp (int)	
ResultSet.getTimestamp (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getTimestamp Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Timestamp getTimestamp(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getTimestamp Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.Timestamp getTimestamp(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getUnicodeStream Method

Overload List

Name	Description
ResultSet.getUnicodeStream (int)	
ResultSet.getUnicodeStream (String)	

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getUnicodeStream Method (Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.io.InputStream getUnicodeStream(
    int columnIndex) throws java.sql.SQLException;
```

Parameters

columnIndex

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getUnicodeStream Method (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.io.InputStream getUnicodeStream(
    java.lang.String columnName) throws java.sql.SQLException;
```

Parameters

columnName

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.getWarnings Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.SQLWarning getWarnings() throws java.sql.SQLException;
```

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.next Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean next() throws java.sql.SQLException;
```

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSet.wasNull Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean wasNull() throws java.sql.SQLException;
```

See Also

Reference

ResultSet Interface

Concepts

ResultSet Members

java.sql Package

ResultSetMetaData Interface

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public interface java.sql.ResultSetMetaData
```

See Also

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData Members

The following tables list the members exposed by the ResultSetMetaData type.

Public Fields

Name	Description
columnNoNulls	
columnNullable	
columnNullableUnknown	

Public Methods

Name	Description
getCatalogName	
getColumnCount	
getColumnDisplaySize	
getColumnLabel	
getColumnName	
getColumnType	
getColumnTypeName	
getPrecision	
getScale	
getSchemaName	
getTableName	
isAutoIncrement	
isCaseSensitive	
isCurrency	
isDefinitelyWritable	
isNullable	
isReadOnly	
isSearchable	
isSigned	

isWritable	

See Also

Reference

ResultSetMetaData Interface

Concepts

java.sql Package

ResultSetMetaData Fields

Public Fields

Name	Description
columnNoNulls	
columnNullable	
columnNullableUnknown	

See Also

Reference

ResultSetMetaData Interface

Concepts

java.sql Package

ResultSetMetaData.columnNoNulls Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int columnNoNulls;
```

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.columnNullable Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int columnNullable;
```

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.columnNullableUnknown Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int columnNullableUnknown;
```

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData Methods

Public Methods

Name	Description
getCatalogName	
getColumnCount	
getColumnDisplaySize	
getColumnLabel	
getColumnName	
getColumnType	
getColumnTypeName	
getPrecision	
getScale	
getSchemaName	
getTableName	
isAutoIncrement	
isCaseSensitive	
isCurrency	
isDefinitelyWritable	
isNullable	
isReadOnly	
isSearchable	
isSigned	
isWritable	

See Also

Reference

ResultSetMetaData Interface

Concepts

java.sql Package

ResultSetMetaData.getCatalogName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getCatalogName(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getColumnCount Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getColumnCount() throws java.sql.SQLException;
```

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getColumnDisplaySize Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getColumnDisplaySize(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getColumnLabel Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getColumnLabel(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getColumnName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getColumnName(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getColumnType Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getColumnType(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getColumnTypeName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getColumnTypeName(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getPrecision Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getPrecision(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getScale Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getScale(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getSchemaName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getSchemaName(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.getTableName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.String getTableName(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.isAutoIncrement Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isAutoIncrement(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.isCaseSensitive Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isCaseSensitive(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.isCurrency Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isCurrency(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.isDefinitelyWritable Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isDefinitelyWritable(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.isNullable Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int isNullable(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.isReadOnly Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isReadOnly(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.isSearchable Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isSearchable(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.isSigned Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isSigned(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

ResultSetMetaData.isWritable Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isWritable(
    int column) throws java.sql.SQLException;
```

Parameters

column

See Also

Reference

ResultSetMetaData Interface

Concepts

ResultSetMetaData Members

java.sql Package

SQLException Class

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.SQLException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.sql.SQLException

java.sql.SQLWarning

See Also

Concepts

SQLException Members

java.sql Package

SQLException Members

The following tables list the members exposed by the SQLException type.

Public Constructors

Name	Description
SQLException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
getErrorCode	
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
getNextException	
GetObjectData	Overridden.
getSQLState	
getClass	(inherited from Object)

clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
setNextException	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SQLException Class

Concepts

java.sql Package

SQLException Constructor

Overload List

Name	Description
SQLException ()	
SQLException (String)	
SQLException (SerializationInfo, StreamingContext)	
SQLException (String, Exception)	
SQLException (String, String)	
SQLException (String, String, int)	
SQLException (String, String, int, Exception)	

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException Constructor ()

Initializes a new instance of the SQLException Class .

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException Constructor (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLException(
    java.lang.String reason);
```

Parameters

reason

See Also
Reference
SQLException Class
Concepts
SQLException Members
java.sql Package

SQLException Constructor (SerializationInfo, StreamingContext)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
protected java.sql.SQLException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException Constructor (String, Exception)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also
Reference
SQLException Class
Concepts
SQLException Members
java.sql Package

SQLException Constructor (String, String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLException(
    java.lang.String reason,
    java.lang.String SQLState);
```

Parameters

reason

SQLState

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException Constructor (String, String, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLException(
    java.lang.String reason,
    java.lang.String SQLState,
    int vendorCode);
```

Parameters

reason

SQLState

vendorCode

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException Constructor (String, String, Int32, Exception)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLException(
    java.lang.String reason,
    java.lang.String SQLState,
    int vendorCode,
    System.Exception inner);
```

Parameters

reason

SQLState

vendorCode

inner

See Also
Reference
SQLException Class
Concepts
SQLException Members
java.sql Package

SQLException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
getErrorCode	
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
getNextException	
GetObjectData	Overridden.
getSQLState	
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
setNextException	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SQLException Class

Concepts

java.sql Package

SQLException.getErrorCode Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getErrorCode();
```

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException.getNextException Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLException getNextException();
```

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException.GetObjectData Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException.getSQLState Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getSQLState();
```

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException.setNextException Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public synchronized void setNextException(
    java.sql.SQLException ex);
```

Parameters

ex

See Also
Reference
SQLException Class
Concepts
SQLException Members
java.sql Package

SQLException.toString Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

SQLException Class

Concepts

SQLException Members

java.sql Package

SQLException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

SQLException Class

Concepts

java.sql Package

SQLWarning Class

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.SQLWarning
    extends java.sql.SQLException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.sql.SQLException

java.sql.SQLWarning

java.sql.DataTruncation

See Also

Concepts

SQLWarning Members

java.sql Package

SQLWarning Members

The following tables list the members exposed by the SQLWarning type.

Public Constructors

Name	Description
SQLWarning	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
getErrorCode	(inherited from SQLException)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
getNextException	(inherited from SQLException)
getNextWarning	
GetObjectData	(inherited from SQLException)
getSQLState	(inherited from SQLException)

getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
setNextException	(inherited from SQLException)
setNextWarning	
toString	(inherited from SQLException)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SQLWarning Class

Concepts

java.sql Package

SQLWarning Constructor

Overload List

Name	Description
SQLWarning ()	
SQLWarning (String)	
SQLWarning (SerializationInfo, StreamingContext)	
SQLWarning (String, Exception)	
SQLWarning (String, String)	
SQLWarning (String, String, int)	
SQLWarning (String, String, int, Exception)	

See Also

Reference

SQLWarning Class

Concepts

SQLWarning Members

java.sql Package

SQLWarning Constructor ()

Initializes a new instance of the SQLWarning Class .

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLWarning();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

SQLWarning Class

Concepts

SQLWarning Members

java.sql Package

SQLWarning Constructor (String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLWarning(
    java.lang.String reason);
```

Parameters

reason

See Also

Reference

SQLWarning Class

Concepts

SQLWarning Members

java.sql Package

SQLWarning Constructor (SerializationInfo, StreamingContext)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
protected java.sql.SQLWarning(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also
Reference
SQLWarning Class
Concepts
SQLWarning Members
java.sql Package

SQLWarning Constructor (String, Exception)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLWarning(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also

Reference

SQLWarning Class

Concepts

SQLWarning Members

java.sql Package

SQLWarning Constructor (String, String)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLWarning(
    java.lang.String reason,
    java.lang.String SQLState);
```

Parameters

reason

SQLState

See Also

Reference

SQLWarning Class

Concepts

SQLWarning Members

java.sql Package

SQLWarning Constructor (String, String, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLWarning(
    java.lang.String reason,
    java.lang.String SQLState,
    int vendorCode);
```

Parameters

reason

SQLState

vendorCode

See Also

Reference

SQLWarning Class

Concepts

SQLWarning Members

java.sql Package

SQLWarning Constructor (String, String, Int32, Exception)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLWarning(
    java.lang.String reason,
    java.lang.String SQLState,
    int vendorCode,
    System.Exception inner);
```

Parameters

reason

SQLState

vendorCode

inner

See Also

Reference

SQLWarning Class

Concepts

SQLWarning Members

java.sql Package

SQLWarning Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
getErrorCode	(inherited from SQLException)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
getNextException	(inherited from SQLException)
getNextWarning	
GetObjectData	(inherited from SQLException)
getSQLState	(inherited from SQLException)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
setNextException	(inherited from SQLException)
setNextWarning	
toString	(inherited from SQLException)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SQLWarning Class

Concepts

java.sql Package

SQLWarning.getNextWarning Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.SQLWarning getNextWarning();
```

See Also

Reference

SQLWarning Class

Concepts

SQLWarning Members

java.sql Package

SQLWarning.setNextWarning Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public synchronized void setNextWarning(
    java.sql.SQLWarning nwg);
```

Parameters

nwg

See Also

Reference

SQLWarning Class

Concepts

SQLWarning Members

java.sql Package

SQLWarning Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

SQLWarning Class

Concepts

java.sql Package

Statement Interface

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public interface java.sql.Statement
```

See Also

Concepts

Statement Members

java.sql Package

Statement Members

The following tables list the members exposed by the Statement type.

Public Methods

Name	Description
cancel	
clearWarnings	
close	
execute	
executeQuery	
executeUpdate	
getMaxFieldSize	
getMaxRows	
getMoreResults	
getQueryTimeout	
getResultSet	
getUpdateCount	
getWarnings	
setCursorName	
setEscapeProcessing	
setMaxFieldSize	
setMaxRows	
setQueryTimeout	

See Also

Reference

Statement Interface

Concepts

java.sql Package

Statement Methods

Public Methods

Name	Description
cancel	
clearWarnings	
close	
execute	
executeQuery	
executeUpdate	
getMaxFieldSize	
getMaxRows	
getMoreResults	
getQueryTimeout	
getResultSet	
getUpdateCount	
getWarnings	
setCursorName	
setEscapeProcessing	
setMaxFieldSize	
setMaxRows	
setQueryTimeout	

See Also

Reference

Statement Interface

Concepts

java.sql Package

Statement.cancel Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void cancel() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.clearWarnings Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void clearWarnings() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.close Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void close() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.execute Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean execute(
    java.lang.String sql) throws java.sql.SQLException;
```

Parameters

sql

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.executeQuery Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet executeQuery(
    java.lang.String sql) throws java.sql.SQLException;
```

Parameters

sql

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.executeUpdate Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int executeUpdate(
    java.lang.String sql) throws java.sql.SQLException;
```

Parameters

sql

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.getMaxFieldSize Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxFieldSize() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.getMaxRows Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaxRows() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.getMoreResults Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean getMoreResults() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.getQueryTimeout Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getQueryTimeout() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.getResultSet Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.ResultSet getResultSet() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.getUpdateCount Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract int getUpdateCount() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.getWarnings Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract java.sql.SQLWarning getWarnings() throws java.sql.SQLException;
```

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.setCursorName Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setCursorName(
    java.lang.String name) throws java.sql.SQLException;
```

Parameters

name

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.setEscapeProcessing Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setEscapeProcessing(
    boolean enable) throws java.sql.SQLException;
```

Parameters

enable

See Also
Reference
Statement Interface
Concepts
Statement Members
java.sql Package

Statement.setMaxFieldSize Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setMaxFieldSize(
    int maxFieldSize) throws java.sql.SQLException;
```

Parameters

maxFieldSize

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.setMaxRows Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setMaxRows(
    int maxRows) throws java.sql.SQLException;
```

Parameters

maxRows

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Statement.setQueryTimeout Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public abstract void setQueryTimeout(
    int queryTimeout) throws java.sql.SQLException;
```

Parameters

queryTimeout

See Also

Reference

Statement Interface

Concepts

Statement Members

java.sql Package

Time Class

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.Time
    extends java.util.Date
```

Inheritance Hierarchy

java.lang.Object

  java.util.Date

   java.sql.Time

See Also

Concepts

Time Members

java.sql Package

Time Members

The following tables list the members exposed by the Time type.

Public Constructors

Name	Description
Time	Overloaded.

Public Methods

Name	Description
after	Determines whether the Date object is after a given date. (inherited from Date)
before	Determines whether the Date object is before a given date. (inherited from Date)
compareTo	Compares two dates. (inherited from Date)
equals	Determines whether two Date objects are equal. (inherited from Date)
getDate	Gets a value indicating the date. (inherited from Date)
getDay	Gets a value indicating the day of the week. (inherited from Date)
hashCode	Generates a hash of the date. (inherited from Date)
getHours	Overridden.
getMinutes	Overridden.
getMonth	Gets a value indicating the month. (inherited from Date)
getSeconds	Overridden.
getTime	Gets a value indicating the time in milliseconds since epoch. (inherited from Date)
getTimezoneOffset	Gets a value indicating the time zone offset in milliseconds. (inherited from Date)
getClass	(inherited from Object)
getYear	Gets a value indicating the year. (inherited from Date)
clone	(inherited from Date)
setDate	Sets a value indicating the date. (inherited from Date)
setHours	Overridden.
setMinutes	Overridden.
setMonth	Overridden.

setSeconds	Overridden.
setTime	Sets the time to the value represented in milliseconds since epoch. (inherited from Date)
setYear	Sets a value indicating the year. (inherited from Date)
ToBoolean	Converts an instance of a Date object to a Boolean. (inherited from Date)
ToByte	Converts an instance of a Date object to a Byte. (inherited from Date)
ToChar	Converts an instance of a Date object to a Char. (inherited from Date)
ToDateTime	Converts an instance of a Date object to a DateTime. (inherited from Date)
ToDecimal	Converts an instance of a Date object to a Decimal. (inherited from Date)
ToDouble	Converts an instance of a Date object to a Double. (inherited from Date)
toGMTString	Displays a value representing the date/time of a Date object in GMT. (inherited from Date)
ToInt16	Converts an instance of a Date object to an Int16. (inherited from Date)
ToInt32	Converts an instance of a Date object to an Int32. (inherited from Date)
ToInt64	Converts an instance of a Date object to an Int64. (inherited from Date)
toLocaleString	Displays a value representing the date/time of a Date object displayed in the format of current locale. (inherited from Date)
ToSByte	Converts an instance of a Date object to a SByte. (inherited from Date)
ToSingle	Converts an instance of a Date object to a Single. (inherited from Date)
toString	Overloaded. Overridden.
ToType	Converts an instance of a Date object to a given Type. (inherited from Date)
ToUInt16	Converts an instance of a Date object to an UInt16. (inherited from Date)
ToUInt32	Converts an instance of a Date object to an UInt32. (inherited from Date)
ToUInt64	Converts an instance of a Date object to an UInt64. (inherited from Date)
valueOf	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Time Class

Concepts

java.sql Package

Time Constructor

Overload List

Name	Description
Time (long)	
Time (int, int, int)	

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time Constructor (Int64)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.Time(
    long msSinceEpoch);
```

Parameters
msSinceEpoch

See Also
Reference
Time Class
Concepts
Time Members
java.sql Package

Time Constructor (Int32, Int32, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.Time(
    int hour,
    int minute,
    int second);
```

Parameters

hour

minute

second

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time Methods

Public Methods

Name	Description
after	Determines whether the Date object is after a given date. (inherited from Date)
before	Determines whether the Date object is before a given date. (inherited from Date)
compareTo	Compares two dates. (inherited from Date)
equals	Determines whether two Date objects are equal. (inherited from Date)
getDate	Gets a value indicating the date. (inherited from Date)
getDay	Gets a value indicating the day of the week. (inherited from Date)
hashCode	Generates a hash of the date. (inherited from Date)
getHours	Overridden.
getMinutes	Overridden.
getMonth	Gets a value indicating the month. (inherited from Date)
getSeconds	Overridden.
getTime	Gets a value indicating the time in milliseconds since epoch. (inherited from Date)
getTimezoneOffset	Gets a value indicating the time zone offset in milliseconds. (inherited from Date)
getClass	(inherited from Object)
getYear	Gets a value indicating the year. (inherited from Date)
clone	(inherited from Date)
setDate	Sets a value indicating the date. (inherited from Date)
setHours	Overridden.
setMinutes	Overridden.
setMonth	Overridden.
setSeconds	Overridden.
setTime	Sets the time to the value represented in milliseconds since epoch. (inherited from Date)
setYear	Sets a value indicating the year. (inherited from Date)

ToBoolean	Converts an instance of a Date object to a Boolean. (inherited from Date)
ToByte	Converts an instance of a Date object to a Byte. (inherited from Date)
ToChar	Converts an instance of a Date object to a Char. (inherited from Date)
ToDateTime	Converts an instance of a Date object to a DateTime. (inherited from Date)
ToDecimal	Converts an instance of a Date object to a Decimal. (inherited from Date)
ToDouble	Converts an instance of a Date object to a Double. (inherited from Date)
toGMTString	Displays a value representing the date/time of a Date object in GMT. (inherited from Date)
ToInt16	Converts an instance of a Date object to an Int16. (inherited from Date)
ToInt32	Converts an instance of a Date object to an Int32. (inherited from Date)
ToInt64	Converts an instance of a Date object to an Int64. (inherited from Date)
toLocaleString	Displays a value representing the date/time of a Date object displayed in the format of current locale. (inherited from Date)
ToSByte	Converts an instance of a Date object to a SByte. (inherited from Date)
ToSingle	Converts an instance of a Date object to a Single. (inherited from Date)
toString	Overloaded. Overridden.
ToType	Converts an instance of a Date object to a given Type. (inherited from Date)
ToUInt16	Converts an instance of a Date object to an UInt16. (inherited from Date)
ToUInt32	Converts an instance of a Date object to an UInt32. (inherited from Date)
ToUInt64	Converts an instance of a Date object to an UInt64. (inherited from Date)
valueOf	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Time Class

Concepts

java.sql Package

Time.getHours Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getHours();
```

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time.getMinutes Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getMinutes();
```

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time.getSeconds Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getSeconds();
```

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time.setHours Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setHours(
    int hour);
```

Parameters

hour

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time.setMinutes Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setMinutes(
    int minute);
```

Parameters

minute

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time.setMonth Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setMonth(
    int second);
```

Parameters

second

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time.setSeconds Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setSeconds(
    int second);
```

Parameters

second

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time.toString Method

Overload List

Name	Description
Time.ToString ()	
Time.ToString (IFormatProvider)	Converts an instance of a Date object to a String.

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time.toString Method ()

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Time.valueOf Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static java.sql.Time valueOf(
    java.lang.String s);
```

Parameters

s

See Also

Reference

Time Class

Concepts

Time Members

java.sql Package

Timestamp Class

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.Timestamp
    extends java.util.Date
```

Inheritance Hierarchy

java.lang.Object

  java.util.Date

   java.sql.Timestamp

See Also

Concepts

Timestamp Members

java.sql Package

Timestamp Members

The following tables list the members exposed by the Timestamp type.

Public Constructors

Name	Description
Timestamp	Overloaded.

Public Methods

Name	Description
after	Determines whether the Date object is after a given date. (inherited from Date)
before	Determines whether the Date object is before a given date. (inherited from Date)
compareTo	Compares two dates. (inherited from Date)
equals	Determines whether two Date objects are equal. (inherited from Date)
getDate	Gets a value indicating the date. (inherited from Date)
getDay	Gets a value indicating the day of the week. (inherited from Date)
hashCode	Generates a hash of the date. (inherited from Date)
getHours	Gets a value indicating the hours. (inherited from Date)
getMinutes	Gets a value indicating the minutes. (inherited from Date)
getMonth	Gets a value indicating the month. (inherited from Date)
getNanos	
getSeconds	Gets a value indicating the seconds. (inherited from Date)
getTime	Gets a value indicating the time in milliseconds since epoch. (inherited from Date)
getTimezoneOffset	Gets a value indicating the time zone offset in milliseconds. (inherited from Date)
getClass	(inherited from Object)
getYear	Gets a value indicating the year. (inherited from Date)
clone	(inherited from Date)
setDate	Sets a value indicating the date. (inherited from Date)
setHours	Sets a value indicating the hours. (inherited from Date)
setMinutes	Sets a value indicating the minutes. (inherited from Date)

setMonth	Sets a value indicating the month. (inherited from Date)
setNanos	
setSeconds	Sets a value indicating the seconds. (inherited from Date)
setTime	Sets the time to the value represented in milliseconds since epoch. (inherited from Date)
setYear	Sets a value indicating the year. (inherited from Date)
ToBoolean	Converts an instance of a Date object to a Boolean. (inherited from Date)
ToByte	Converts an instance of a Date object to a Byte. (inherited from Date)
ToChar	Converts an instance of a Date object to a Char. (inherited from Date)
ToDateTime	Converts an instance of a Date object to a DateTime. (inherited from Date)
ToDecimal	Converts an instance of a Date object to a Decimal. (inherited from Date)
ToDouble	Converts an instance of a Date object to a Double. (inherited from Date)
toGMTString	Displays a value representing the date/time of a Date object in GMT. (inherited from Date)
ToInt16	Converts an instance of a Date object to an Int16. (inherited from Date)
ToInt32	Converts an instance of a Date object to an Int32. (inherited from Date)
ToInt64	Converts an instance of a Date object to an Int64. (inherited from Date)
toLocaleString	Displays a value representing the date/time of a Date object displayed in the format of current locale. (inherited from Date)
ToSByte	Converts an instance of a Date object to a SByte. (inherited from Date)
ToSingle	Converts an instance of a Date object to a Single. (inherited from Date)
toString	Overloaded. Overridden.
ToType	Converts an instance of a Date object to a given Type. (inherited from Date)
ToUInt16	Converts an instance of a Date object to an UInt16. (inherited from Date)
ToUInt32	Converts an instance of a Date object to an UInt32. (inherited from Date)
ToUInt64	Converts an instance of a Date object to an UInt64. (inherited from Date)
valueOf	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Timestamp Class

Concepts

java.sql Package

Timestamp Constructor

Overload List

Name	Description
Timestamp (long)	
Timestamp (int, int, int, int, int, int, int)	

See Also

Reference

Timestamp Class

Concepts

Timestamp Members

java.sql Package

Timestamp Constructor (Int64)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.Timestamp(
    long msSinceEpoch);
```

Parameters

msSinceEpoch

See Also

Reference

Timestamp Class

Concepts

Timestamp Members

java.sql Package

Timestamp Constructor (Int32, Int32, Int32, Int32, Int32, Int32, Int32)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.Timestamp(
    int year,
    int month,
    int date,
    int hour,
    int minute,
    int second,
    int nano);
```

Parameters

year

month

date

hour

minute

second

nano

See Also

Reference

Timestamp Class

Concepts

Timestamp Members

java.sql Package

Timestamp Methods

Public Methods

Name	Description
after	Determines whether the Date object is after a given date. (inherited from Date)
before	Determines whether the Date object is before a given date. (inherited from Date)
compareTo	Compares two dates. (inherited from Date)
equals	Determines whether two Date objects are equal. (inherited from Date)
getDate	Gets a value indicating the date. (inherited from Date)
getDay	Gets a value indicating the day of the week. (inherited from Date)
hashCode	Generates a hash of the date. (inherited from Date)
getHours	Gets a value indicating the hours. (inherited from Date)
getMinutes	Gets a value indicating the minutes. (inherited from Date)
getMonth	Gets a value indicating the month. (inherited from Date)
getNanos	
getSeconds	Gets a value indicating the seconds. (inherited from Date)
getTime	Gets a value indicating the time in milliseconds since epoch. (inherited from Date)
getTimezoneOffset	Gets a value indicating the time zone offset in milliseconds. (inherited from Date)
getClass	(inherited from Object)
getYear	Gets a value indicating the year. (inherited from Date)
clone	(inherited from Date)
setDate	Sets a value indicating the date. (inherited from Date)
setHours	Sets a value indicating the hours. (inherited from Date)
setMinutes	Sets a value indicating the minutes. (inherited from Date)
setMonth	Sets a value indicating the month. (inherited from Date)
setNanos	
setSeconds	Sets a value indicating the seconds. (inherited from Date)

setTime	Sets the time to the value represented in milliseconds since epoch. (inherited from Date)
setYear	Sets a value indicating the year. (inherited from Date)
ToBoolean	Converts an instance of a Date object to a Boolean. (inherited from Date)
ToByte	Converts an instance of a Date object to a Byte. (inherited from Date)
ToChar	Converts an instance of a Date object to a Char. (inherited from Date)
ToDateTime	Converts an instance of a Date object to a DateTime. (inherited from Date)
ToDecimal	Converts an instance of a Date object to a Decimal. (inherited from Date)
ToDouble	Converts an instance of a Date object to a Double. (inherited from Date)
toGMTString	Displays a value representing the date/time of a Date object in GMT. (inherited from Date)
ToInt16	Converts an instance of a Date object to an Int16. (inherited from Date)
ToInt32	Converts an instance of a Date object to an Int32. (inherited from Date)
ToInt64	Converts an instance of a Date object to an Int64. (inherited from Date)
toLocaleString	Displays a value representing the date/time of a Date object displayed in the format of current locale. (inherited from Date)
ToSByte	Converts an instance of a Date object to a SByte. (inherited from Date)
ToSingle	Converts an instance of a Date object to a Single. (inherited from Date)
toString	Overloaded. Overridden.
ToType	Converts an instance of a Date object to a given Type. (inherited from Date)
ToUInt16	Converts an instance of a Date object to an UInt16. (inherited from Date)
ToUInt32	Converts an instance of a Date object to an UInt32. (inherited from Date)
ToUInt64	Converts an instance of a Date object to an UInt64. (inherited from Date)
valueOf	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Timestamp Class

Concepts

java.sql Package

Timestamp.getNanos Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getNanos();
```

See Also

Reference

Timestamp Class

Concepts

Timestamp Members

java.sql Package

Timestamp.setNanos Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setNanos(
    int nano);
```

Parameters

nano

See Also

Reference

Timestamp Class

Concepts

Timestamp Members

java.sql Package

Timestamp.toString Method

Overload List

Name	Description
Timestamp.ToString ()	
Timestamp.ToString (IFormatProvider)	Converts an instance of a Date object to a String.

See Also

Reference

Timestamp Class

Concepts

Timestamp Members

java.sql Package

Timestamp.toString Method ()

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

Timestamp Class

Concepts

Timestamp Members

java.sql Package

Timestamp.valueOf Method

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static java.sql.Timestamp valueOf(
    java.lang.String s);
```

Parameters

s

See Also

Reference

Timestamp Class

Concepts

Timestamp Members

java.sql Package

Types Class

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.Types
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.sql.Types

See Also

Concepts

Types Members

java.sql Package

Types Members

The following tables list the members exposed by the Types type.

Public Constructors

Name	Description
Types	

Public Fields

Name	Description
BIGINT	
BINARY	
BIT	
CHAR	
DATE	
DECIMAL	
DOUBLE	
FLOAT	
INTEGER	
LONGVARBINARY	
LONGVARCHAR	
NULL	
NUMERIC	
OTHER	
REAL	
SMALLINT	
TIME	
TIMESTAMP	
TINYINT	
VARBINARY	
VARCHAR	

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Types Class

Concepts

java.sql Package

Types Fields

Public Fields

Name	Description
BIGINT	
BINARY	
BIT	
CHAR	
DATE	
DECIMAL	
DOUBLE	
FLOAT	
INTEGER	
LONGVARBINARY	
LONGVARCHAR	
NULL	
NUMERIC	
OTHER	
REAL	
SMALLINT	
TIME	
TIMESTAMP	
TINYINT	
VARBINARY	
VARCHAR	

See Also

Reference

Types Class

Concepts

java.sql Package

Types.BIGINT Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int BIGINT;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.BINARY Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int BINARY;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.BIT Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int BIT;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.CHAR Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int CHAR;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.DATE Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int DATE;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.DECIMAL Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int DECIMAL;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.DOUBLE Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int DOUBLE;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.FLOAT Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int FLOAT;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.INTEGER Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int INTEGER;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.LONGVARBINARY Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int LONGVARBINARY;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.LONGVARCHAR Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int LONGVARCHAR;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.NULL Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int NULL;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.NUMERIC Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int NUMERIC;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.OTHER Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int OTHER;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.REAL Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int REAL;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.SMALLINT Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int SMALLINT;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.TIME Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int TIME;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.TIMESTAMP Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int TIMESTAMP;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.TINYINT Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int TINYINT;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.VARBINARY Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int VARBINARY;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types.VARCHAR Field

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public static final int VARCHAR;
```

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types Constructor

Initializes a new instance of the Types Class .

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.Types();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Types Class

Concepts

Types Members

java.sql Package

Types Methods

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Types Class

Concepts

java.sql Package

java.text

Contains the classes used for processing text, numbers and dates.

Classes

Class	Description
Collator	Contains methods used to sort and arrange strings of Unicode characters based on their decomposition and strength.
DateFormat	Contains methods and properties used to format a Date object as a string and to parse a string into a Date object.
DecimalFormat	Contains methods and properties used to format a number as a string and to parse a string into a number. This class provides a default implementation of the abstract class NumberFormat.
Format	An abstract class used as the base class for the rest of the format classes in the java.text package.
MessageFormat	Contains methods and properties used to format a message as a string and to parse a string into a message. A message is simply a collection of objects that are formatted according to their type.
NumberFormat	Contains methods and properties used to format a number as a string and to parse a string into a number.
SimpleDateFormat	Contains methods and properties used to format a Date object as a string and to parse a string into a Date object.

BreakIterator Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.text.BreakIterator
    extends java.lang.Object
    implements java.lang.Cloneable, java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

java.text.BreakIterator

See Also

Concepts

BreakIterator Members

java.text Package

BreakIterator Members

The following tables list the members exposed by the BreakIterator type.

Public Constructors

Name	Description
BreakIterator	

Public Fields

Name	Description
DONE	

Public Methods

Name	Description
current	
equals	(inherited from Object)
first	
following	
getAvailableLocales	
getCharacterInstance	Overloaded.
hashCode	(inherited from Object)
getLineInstance	Overloaded.
getSentenceInstance	Overloaded.
getText	
getClass	(inherited from Object)
getWordInstance	Overloaded.
last	
MemberwiseClone	Performs a shallow copy of the members.
next	Overloaded.
previous	
setText	Overloaded.
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BreakIterator Class

Concepts

java.text Package

BreakIterator Fields

Public Fields

Name	Description
DONE	

See Also

Reference

BreakIterator Class

Concepts

java.text Package

BreakIterator.DONE Field

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int DONE;
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator Constructor

Initializes a new instance of the BreakIterator Class .

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
protected java.text.BreakIterator();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator Methods

Public Methods

Name	Description
current	
equals	(inherited from Object)
first	
following	
getAvailableLocales	
getCharacterInstance	Overloaded.
hashCode	(inherited from Object)
getLineInstance	Overloaded.
getSentenceInstance	Overloaded.
getText	
getClass	(inherited from Object)
getWordInstance	Overloaded.
last	
MemberwiseClone	Performs a shallow copy of the members.
next	Overloaded.
previous	
setText	Overloaded.
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BreakIterator Class

Concepts

java.text Package

BreakIterator.clone Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.current Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int current();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.first Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int first();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.following Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int following(
    int pos);
```

Parameters

pos

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getAvailableLocales Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.util.Locale[] getAvailableLocales();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getCharacterInstance Method

Overload List

Name	Description
BreakIterator.getCharacterInstance ()	
BreakIterator.getCharacterInstance (Locale)	

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getCharacterInstance Method ()

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.BreakIterator getCharacterInstance();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getCharacterInstance Method (Locale)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.BreakIterator getCharacterInstance(
    java.util.Locale loc);
```

Parameters

loc

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getLineInstance Method

Overload List

Name	Description
BreakIterator.getLineInstance ()	
BreakIterator.getLineInstance (Locale)	

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getLineInstance Method ()

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.BreakIterator getLineInstance();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getLineInstance Method (Locale)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.BreakIterator getLineInstance(
    java.util.Locale loc);
```

Parameters

loc

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getSentenceInstance Method

Overload List

Name	Description
BreakIterator.getSentenceInstance ()	
BreakIterator.getSentenceInstance (Locale)	

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getSentenceInstance Method ()

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.BreakIterator getSentenceInstance();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getSentenceInstance Method (Locale)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.BreakIterator getSentenceInstance(
    java.util.Locale loc);
```

Parameters

loc

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getText Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.text.CharacterIterator getText();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getWordInstance Method

Overload List

Name	Description
BreakIterator.getWordInstance ()	
BreakIterator.getWordInstance (Locale)	

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getWordInstance Method ()

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.BreakIterator getWordInstance();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.getWordInstance Method (Locale)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.BreakIterator getWordInstance(
    java.util.Locale loc);
```

Parameters

loc

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.last Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int last();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.next Method

Overload List

Name	Description
BreakIterator.next ()	
BreakIterator.next (int)	

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.next Method ()

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int next();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.next Method (Int32)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int next(
    int n);
```

Parameters

n

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.previous Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int previous();
```

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.setText Method

Overload List

Name	Description
BreakIterator.setText (CharacterIterator)	
BreakIterator.setText (String)	

See Also

Reference

BreakIterator Class

Concepts

BreakIterator Members

java.text Package

BreakIterator.setText Method (CharacterIterator)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract void setText(
    java.text.CharacterIterator newText);
```

Parameters

newText

See Also
Reference
BreakIterator Class
Concepts
BreakIterator Members
java.text Package

BreakIterator.setText Method (String)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setText(
    java.lang.String newText);
```

Parameters
newText

See Also
Reference
BreakIterator Class
Concepts
BreakIterator Members
java.text Package

CharacterIterator Interface

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public interface java.text.CharacterIterator
    extends java.lang.Cloneable
```

See Also

Concepts

CharacterIterator Members

java.text Package

CharacterIterator Members

The following tables list the members exposed by the CharacterIterator type.

Public Fields

Name	Description
DONE	

Public Methods

Name	Description
current	
first	
getBeginIndex	
getEndIndex	
getIndex	
last	
clone	
next	
previous	
setIndex	

See Also

Reference

CharacterIterator Interface

Concepts

java.text Package

CharacterIterator Fields

Public Fields

Name	Description
DONE	

See Also

Reference

CharacterIterator Interface

Concepts

java.text Package

CharacterIterator.DONE Field

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final char DONE;
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator Methods

Public Methods

Name	Description
current	
first	
getBeginIndex	
getEndIndex	
getIndex	
last	
clone	
next	
previous	
setIndex	

See Also

Reference

CharacterIterator Interface

Concepts

java.text Package

CharacterIterator.clone Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object clone();
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator.current Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract char current();
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator.first Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract char first();
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator.getBeginIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int getBeginIndex();
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator.getEndIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int getEndIndex();
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator.getIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int getIndex();
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator.last Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract char last();
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator.next Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract char next();
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator.previous Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract char previous();
```

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

CharacterIterator.setIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract char setIndex(
    int idx);
```

Parameters

idx

See Also

Reference

CharacterIterator Interface

Concepts

CharacterIterator Members

java.text Package

ChoiceFormat Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public class java.text.ChoiceFormat
    extends java.text.NumberFormat
```

Inheritance Hierarchy

java.lang.Object

  java.text.Format

    java.text.NumberFormat

      java.text.ChoiceFormat

See Also

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat Members

The following tables list the members exposed by the ChoiceFormat type.

Public Constructors

Name	Description
ChoiceFormat	Overloaded.

Public Methods

Name	Description
applyPattern	
equals	Overridden.
format	Overloaded.
getFormats	
hashCode	Overridden.
getLimits	
getMaximumFractionDigits	Gets a value representing the maximum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
getMaximumIntegerDigits	Gets a value representing the maximum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
getMinimumFractionDigits	Gets a value representing the minimum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
getMinimumIntegerDigits	Gets a value representing the minimum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
getClass	(inherited from Object)
isGroupingUsed	Determines whether digits are being grouped together. (inherited from NumberFormat)
isParseIntegerOnly	Determines whether only the integer portion of the number is being parsed or if the fractional part is also included. (inherited from NumberFormat)
clone	
nextDouble	Overloaded.
parse	Overloaded.
parseObject	Overloaded. Parses a string into an Object. (inherited from NumberFormat)
previousDouble	
setChoices	

setGroupingUsed	Sets a value determining whether digits are being grouped together. (inherited from NumberFormat)
setMaximumFractionDigits	Sets the value representing the maximum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
setMaximumIntegerDigits	Sets the value representing the maximum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
setMinimumFractionDigits	Sets the value representing the minimum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
setMinimumIntegerDigits	Sets the value representing the minimum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
setParseIntegerOnly	Sets a value determining whether only the integer portion of the number is being parsed or if the fractional part is also included. (inherited from NumberFormat)
toPattern	
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ChoiceFormat Class

Concepts

java.text Package

ChoiceFormat Constructor

Overload List

Name	Description
ChoiceFormat (String)	
ChoiceFormat (double[], String[])	

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat Constructor (String)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.ChoiceFormat(
    java.lang.String newPattern);
```

Parameters

newPattern

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat Constructor (Double[ ], String[ ])

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.ChoiceFormat(
    double[] limits,
    java.lang.String[] strings);
```

Parameters

limits

strings

See Also
Reference
ChoiceFormat Class
Concepts
ChoiceFormat Members
java.text Package

ChoiceFormat Methods

Public Methods

Name	Description
applyPattern	
equals	Overridden.
format	Overloaded.
getFormats	
hashCode	Overridden.
getLimits	
getMaximumFractionDigits	Gets a value representing the maximum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
getMaximumIntegerDigits	Gets a value representing the maximum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
getMinimumFractionDigits	Gets a value representing the minimum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
getMinimumIntegerDigits	Gets a value representing the minimum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
getClass	(inherited from Object)
isGroupingUsed	Determines whether digits are being grouped together. (inherited from NumberFormat)
isParseIntegerOnly	Determines whether only the integer portion of the number is being parsed or if the fractional part is also included. (inherited from NumberFormat)
clone	
nextDouble	Overloaded.
parse	Overloaded.
parseObject	Overloaded. Parses a string into an Object. (inherited from NumberFormat)
previousDouble	
setChoices	
setGroupingUsed	Sets a value determining whether digits are being grouped together. (inherited from NumberFormat)
setMaximumFractionDigits	Sets the value representing the maximum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)

setMaximumIntegerDigits	Sets the value representing the maximum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
setMinimumFractionDigits	Sets the value representing the minimum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
setMinimumIntegerDigits	Sets the value representing the minimum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
setParseIntegerOnly	Sets a value determining whether only the integer portion of the number is being parsed or if the fractional part is also included. (inherited from NumberFormat)
toPattern	
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ChoiceFormat Class

Concepts

java.text Package

ChoiceFormat.applyPattern Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void applyPattern(
    java.lang.String newPattern);
```

Parameters
newPattern

See Also
Reference
ChoiceFormat Class
Concepts
ChoiceFormat Members
java.text Package

ChoiceFormat.clone Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object clone();
```

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.equals Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.format Method

Overload List

Name	Description
ChoiceFormat.format (double)	Formats a double into a string.
ChoiceFormat.format (long)	Formats a long into a string.
ChoiceFormat.format (Object)	Formats an object into a string.
ChoiceFormat.format (double, StringBuffer, FieldPosition)	
ChoiceFormat.format (long, StringBuffer, FieldPosition)	
ChoiceFormat.format (Object, StringBuffer, FieldPosition)	Formats an Object and appends it to a StringBuffer object.

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.format Method (Double, StringBuffer, FieldPosition)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer format(
    double num,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition pos);
```

Parameters

num

appendBuf

pos

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.format Method (Int64,StringBuffer, FieldPosition)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer format(
    long num,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition pos);
```

Parameters

num

appendBuf

pos

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.getFormats Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object[] getFormats();
```

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.hashCode Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.getLimits Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public double[] getLimits();
```

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.nextDouble Method

Overload List

Name	Description
ChoiceFormat.nextDouble (double)	
ChoiceFormat.nextDouble (double, boolean)	

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.nextDouble Method (Double)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final double nextDouble(
    double d);
```

Parameters

d

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.nextDouble Method (Double, Boolean)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static double nextDouble(
    double dblNum,
    boolean next);
```

Parameters

dblNum

next

See Also
Reference
ChoiceFormat Class
Concepts
ChoiceFormat Members
java.text Package

ChoiceFormat.parse Method

Overload List

Name	Description
ChoiceFormat.parse (String)	Parses a string into a number represented by a Number object.
ChoiceFormat.parse (String, ParsePosition)	

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.parse Method (String, ParsePosition)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Number parse(
    java.lang.String sourceStr,
    java.text.ParsePosition pos);
```

Parameters

sourceStr

pos

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.previousDouble Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final double previousDouble(
    double d);
```

Parameters

d

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.setChoices Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setChoices(
    double[] limits,
    java.lang.String[] strings);
```

Parameters

limits

strings

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

ChoiceFormat.toPattern Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toPattern();
```

See Also

Reference

ChoiceFormat Class

Concepts

ChoiceFormat Members

java.text Package

CollationKey Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final class java.text.CollationKey
    extends java.lang.Object
    implements java.lang.Comparable
```

Inheritance Hierarchy

java.lang.Object

java.text.CollationKey

See Also

Concepts

CollationKey Members

java.text Package

CollationKey Members

The following tables list the members exposed by the CollationKey type.

Public Methods

Name	Description
compareTo	Overloaded.
equals	Overridden.
hashCode	Overridden.
getSourceString	
getClass	(inherited from Object)
toByteArray	
toString	Overridden.

Protected Members

Name	Description
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

CollationKey Class

Concepts

java.text Package

CollationKey Methods

Public Methods

Name	Description
compareTo	Overloaded.
equals	Overridden.
hashCode	Overridden.
getSourceString	
getClass	(inherited from Object)
toByteArray	
toString	Overridden.

Protected Methods

Name	Description
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

CollationKey Class

Concepts

java.text Package

CollationKey.compareTo Method

Overload List

Name	Description
CollationKey.compareTo (CollationKey)	
CollationKey.compareTo (Object)	

See Also

Reference

CollationKey Class

Concepts

CollationKey Members

java.text Package

CollationKey.compareTo Method (CollationKey)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.text.CollationKey target);
```

Parameters

target

See Also
Reference
CollationKey Class
Concepts
CollationKey Members
java.text Package

CollationKey.compareTo Method (Object)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

CollationKey Class

Concepts

CollationKey Members

java.text Package

CollationKey.equals Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object target);
```

Parameters

target

See Also

Reference

CollationKey Class

Concepts

CollationKey Members

java.text Package

CollationKey.hashCode Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

CollationKey Class

Concepts

CollationKey Members

java.text Package

CollationKey.getSourceString Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getSourceString();
```

See Also

Reference

CollationKey Class

Concepts

CollationKey Members

java.text Package

CollationKey.toByteArray Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public byte[] toByteArray();
```

See Also

Reference

CollationKey Class

Concepts

CollationKey Members

java.text Package

Collator Class

Contains methods used to sort and arrange strings of Unicode characters based on their decomposition and strength.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.text.Collator
    extends java.lang.Object
    implements java.lang.Cloneable, java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

java.text.Collator

java.text.RuleBasedCollator

See Also

Concepts

Collator Members

java.text Package

Collator Members

Contains methods used to sort and arrange strings of Unicode characters based on their decomposition and strength.

The following tables list the members exposed by the Collator type.

Public Constructors

Name	Description
Collator	Initializes a new instance of a Collator object.

Public Fields

Name	Description
CANONICAL_DECOMPOSITION	The flag used to indicate that all characters are to be decomposed before collation. All precomposed or combined characters are to be decomposed as defined by the Unicode standard.
FULL_DECOMPOSITION	The flag used to indicate that all characters are to be decomposed before collation by doing both canonical and compatibility decomposition as defined by the Unicode standard.
IDENTICAL	The flag used to indicate that two characters are considered equal only if they are equal in their binary values regardless of their Unicode or language properties.
NO_DECOMPOSITION	The flag used to indicate that accented or precomposed characters should not be decomposed before collation.
PRIMARY	The flag used to indicate that two characters are equal if the base characters are equal while accent and case are ignored.
SECONDARY	The flag used to indicate that two characters are equal if their PRIMARY strengths as well as their accents are equal. The case of the character is ignored.
TERTIARY	The flag used to indicate that two characters are equal if their SECONDARY strengths as well as their case are equal.

Public Methods

Name	Description
compare	Compares two strings for equality.
equals	Determines whether two strings are equal.
equals	Overridden. Determines whether two Collator objects are equal.
getAvailableLocales	Gets an array of all Locale objects that are supported by the Collator class.
getCollationKey	Gets an instance of a CollationKey object for the given string.
getDecomposition	Gets the decomposition type being used for collation.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getInstance	Overloaded. Gets an instance of a Collator object.

getStrength	Gets the strength type being used for collation.
getClass	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.
setDecomposition	Sets the decomposition type being used for collation.
setStrength	Sets the strength type being used for collation.
toString	Overridden. Displays a human-readable representation of a Collator object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Collator Class

Concepts

java.text Package

Collator Fields

Public Fields

Name	Description
CANONICAL_DECOMPOSITION	The flag used to indicate that all characters are to be decomposed before collation. All precomposed or combined characters are to be decomposed as defined by the Unicode standard.
FULL_DECOMPOSITION	The flag used to indicate that all characters are to be decomposed before collation by doing both canonical and compatibility decomposition as defined by the Unicode standard.
IDENTICAL	The flag used to indicate that two characters are considered equal only if they are equal in their binary values regardless of their Unicode or language properties.
NO_DECOMPOSITION	The flag used to indicate that accented or precomposed characters should not be decomposed before collation.
PRIMARY	The flag used to indicate that two characters are equal if the base characters are equal while accent and case are ignored.
SECONDARY	The flag used to indicate that two characters are equal if their PRIMARY strengths as well as their accents are equal. The case of the character is ignored.
TERTIARY	The flag used to indicate that two characters are equal if their SECONDARY strengths as well as their case are equal.

See Also

Reference

Collator Class

Concepts

java.text Package

Collator.CANONICAL_DECOMPOSITION Field

The flag used to indicate that all characters are to be decomposed before collation. All precomposed or combined characters are to be decomposed as defined by the Unicode standard.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int CANONICAL_DECOMPOSITION;
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.FULL_DECOMPOSITION Field

The flag used to indicate that all characters are to be decomposed before collation by doing both canonical and compatibility decomposition as defined by the Unicode standard.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int FULL_DECOMPOSITION;
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.IDENTICAL Field

The flag used to indicate that two characters are considered equal only if they are equal in their binary values regardless of their Unicode or language properties.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int IDENTICAL;
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.NO_DECOMPOSITION Field

The flag used to indicate that accented or precomposed characters should not be decomposed before collation.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int NO_DECOMPOSITION;
```

Remarks

Using this flag will block any pre-processing on strings for collation and hence the comparisons may produce unexpected results if accented characters are present and they are desired to be treated as equal.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.PRIMARY Field

The flag used to indicate that two characters are equal if the base characters are equal while accent and case are ignored.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int PRIMARY;
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.SECONDARY Field

The flag used to indicate that two characters are equal if their PRIMARY strengths as well as their accents are equal. The case of the character is ignored.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int SECONDARY;
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.TERTIARY Field

The flag used to indicate that two characters are equal if their SECONDARY strengths as well as their case are equal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int TERTIARY;
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator Constructor

Initializes a new instance of a Collator object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
protected java.text.Collator();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator Methods

Public Methods

Name	Description
compare	Compares two strings for equality.
equals	Determines whether two strings are equal.
equals	Overridden. Determines whether two Collator objects are equal.
getAvailableLocales	Gets an array of all Locale objects that are supported by the Collator class.
getCollationKey	Gets an instance of a CollationKey object for the given string.
getDecomposition	Gets the decomposition type being used for collation.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getInstance	Overloaded. Gets an instance of a Collator object.
getStrength	Gets the strength type being used for collation.
getClass	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.
setDecomposition	Sets the decomposition type being used for collation.
setStrength	Sets the strength type being used for collation.
toString	Overridden. Displays a human-readable representation of a Collator object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Collator Class

Concepts

java.text Package

Collator.clone Method

Creates an instance of a Collator object that is a deep copy of the current Collator object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

Return Value

An instance of a Collator object that is a deep copy of the current Collator object.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.compare Method

Compares two strings for equality.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int compare(
    java.lang.String source,
    java.lang.String target);
```

Parameters

source

The first string to compare for equality.

target

The second string to compare for equality.

Return Value

-1 if source is less than target; 0 if source and target are equal; or 1 if source is greater than target.

Example

The following example compares two strings for equality.

```
// collator_compare.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default instance of a Collator object.
        Collator coll = Collator.getInstance();

        // Compare two strings using a Collator object.
        String a = "StringA";
        String b = "StringB";

        int result = coll.compare(a, b);

        System.out.println(a + "\n" + b + "\n" +
            "Results of compare = " + result);
    }
}

/*
Output:
StringA
StringB
Results of compare = -1
*/
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.equals Method

Determines whether two Collator objects are equal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The Collator object to compare against the current Collator object for equality.

Return Value

true if the two Collator objects are equal; false otherwise.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.equals Method (J#)

Determines whether two Collator objects are equal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The Collator object to compare against the current Collator object for equality.

Return Value

true if the two Collator objects are equal; false otherwise.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.getAvailableLocales Method

Gets an array of all Locale objects that are supported by the Collator class.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.util.Locale[] getAvailableLocales();
```

Return Value

An array of all the Locale objects that are available.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.getCollationKey Method

Gets an instance of a CollationKey object for the given string.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.text.CollationKey getCollationKey(
    java.lang.String source);
```

Parameters

source

The string used as the source for the CollationKey object.

Return Value

An instance of a CollationKey object for the given string.

Example

The following example gets the CollationKey object for an instance of a Collator object.

```
// collator_getcollationkey.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default instance of a Collator object.
        Collator coll = Collator.getInstance();

        // Get the CollationKey for this Collator object.
        CollationKey collKey = coll.getCollationKey("sortKey");
        System.out.println("CollationKey.getSourceString = " +
            collKey.getSourceString());
    }
}

/*
Output:
CollationKey.getSourceString = sortKey
*/
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.getDecomposition Method

Gets the decomposition type being used for collation.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public synchronized int getDecomposition();
```

Return Value

The decomposition type being used for collation. One of NO_DECOMPOSITION, CANONICAL_DECOMPOSITION, or FULL_DECOMPOSITION.

Example

The following example gets the decomposition type being used for collation.

```
// collator_getdecomposition.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default instance of a Collator object.
        Collator coll = Collator.getInstance();

        // Get the decomposition being used for collation.
        int decomp = coll.getDecomposition();
        System.out.println("decomposition = " +
            decomp);
    }
}

/*
Output:
decomposition = 1
*/
```

Remarks

Decomposition defines how characters are to be pre-processed for comparison –- mainly how precomposed characters should be handled or decomposed within strings.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.hashCode Method

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract int hashCode();
```

Return Value

A hash code for the current object.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.getInstance Method

Gets an instance of a Collator object.

Overload List

Name	Description
Collator.getInstance ()	Gets a default instance of a Collator object.
Collator.getInstance (Locale)	Gets an instance of a Collator object for the given Locale.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.getInstance Method ()

Gets a default instance of a Collator object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.text.Collator getInstance();
```

Return Value

A default instance of a Collator object.

Example

The following example demonstrates how to get a default instance of a Collator object.

```
// collator_getinstance.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default instance of a Collator object.
        Collator coll = Collator.getInstance();

        // Get the decomposition being used for collation.
        int decomp = coll.getDecomposition();
        System.out.println("decomposition = " +
            decomp);
    }
}

/*
Output:
decomposition = 1
*/
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.getInstance Method (Locale)

Gets an instance of a Collator object for the given Locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.text.Collator getInstance(
    java.util.Locale loc);
```

Parameters

loc

The Locale to use for collation.

Return Value

An instance of a Collator object for the given Locale.

Example

The following example demonstrates how to get a default instance of a Collator object using the Italy locale.

```
// collator_getinstance_2.jsl

import java.text.*;
import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default instance of a Collator object
        // using the Italy locale.
        Locale loc = new Locale(Locale.ITALY);
        Collator coll = Collator.getInstance(loc);

        // Get the decomposition being used for collation.
        int decomp = coll.getDecomposition();
        System.out.println("decomposition = " +
            decomp);
    }
}

/*
Output:
decomposition = 1
*/
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.getStrength Method

Gets the strength type being used for collation.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public synchronized int getStrength();
```

Return Value

The strength type being used for collation. One of PRIMARY, SECONDARY, TERTIARY, or IDENTICAL.

Example

The following example gets the strength being used for collation.

```
// collator_getstrength.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default instance of a Collator object.
        Collator coll = Collator.getInstance();

        // Get the strength being used for collation.
        int strength = coll.getStrength();
        System.out.println("strength = " +
            strength);
    }
}

/*
Output:
strength = 2
*/
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.setDecomposition Method

Sets the decomposition type being used for collation.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public synchronized void setDecomposition(
    int mode);
```

Parameters

mode

The decomposition type being used for collation. One of NO_DECOMPOSITION, CANONICAL_DECOMPOSITION, or FULL_DECOMPOSITION.

Example

The following example sets the decomposition to FULL_DECOMPOSITION.

```
// collator_setdecomposition.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default instance of a Collator object.
        Collator coll = Collator.getInstance();

        // Set the decomposition to FULL_DECOMPOSITION.
        coll.setDecomposition(Collator.FULL_DECOMPOSITION);

        // Get the decomposition being used for collation.
        int decomp = coll.getDecomposition();
        System.out.println("decomposition = " +
            decomp);
    }
}

/*
Output:
decomposition = 2
*/
```

Remarks

Decomposition defines how characters are to be pre-processed for comparison –- mainly how precomposed characters should be handled or decomposed within strings.

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

Collator.setStrength Method

Sets the strength type being used for collation.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public synchronized void setStrength(
    int strength);
```

Parameters

strength

The strength type being used for collation. One of PRIMARY, SECONDARY, TERTIARY, or IDENTICAL.

Example

The following example sets the strength to IDENTICAL.

```
// collator_setstrength.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default instance of a Collator object.
        Collator coll = Collator.getInstance();

        // Set the strength to IDENTICAL.
        coll.setStrength(Collator.IDENTICAL);

        // Get the strength being used for collation.
        int strength = coll.getStrength();
        System.out.println("strength = " +
            strength);
    }
}

/*
Output:
strength = 3
*/
```

See Also

Reference

Collator Class

Concepts

Collator Members

java.text Package

DateFormat Class

Contains methods and properties used to format a Date object as a string and to parse a string into a Date object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.text.DateFormat
    extends java.text.Format
    implements java.lang.Cloneable
```

Example

The following example demonstrates some common uses of the DateFormat object, such as formatting, parsing, and applying patterns.

```
// dateformat_overview.jsl

import java.text.*;
import java.util.Date;
import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        // Create a SimpleDateFormat object.
        DateFormat sdf = DateFormat.getDateInstance(DateFormat.LONG, Locale.US);

        // Print out the date using the supplied pattern.
        System.out.println("A sample date looks like: " +
            sdf.format(new Date("15-Aug-1947")));
    }
}

/*
Sample output:
A sample date looks like: Friday, August 15, 1947
*/
```

Inheritance Hierarchy

java.lang.Object

  java.text.Format

  java.text.DateFormat

    java.text.SimpleDateFormat

See Also

Concepts

DateFormat Members

java.text Package

DateFormat Members

Contains methods and properties used to format a Date object as a string and to parse a string into a Date object.

The following tables list the members exposed by the DateFormat type.

Public Constructors

Name	Description
DateFormat	Initializes a new instance of a DateFormat object.

Public Fields

Name	Description
AM_PM_FIELD	The value for the style used to indicate that the AM or PM field should be displayed when p rinting a time.
calendar	The underlying Calendar object being used for dates and times.
DATE_FIELD	The value for the style used to indicate that the date field should be displayed when printin g a date.
DAY_OF_WEEK_FIELD	The value for the style used to indicate that the day of the week field should be displayed w hen printing a date.
DAY_OF_WEEK_IN_MONTH_FIELD	The value for the style used to indicate that the day of the week in the month field should b e displayed when printing a date.
DAY_OF_YEAR_FIELD	The value for the style used to indicate that the day of the year field should be displayed wh en printing a date.
DEFAULT	The value for the style used to indicate that the default fields should be displayed when prin ting a date and time.
ERA_FIELD	The value for the style used to indicate that the era field should be displayed when printing a date.
FULL	The value for the style used to indicate that all fields should be displayed when printing a d ate.
HOUR_OF_DAY0_FIELD	The value for the style used to indicate that the zero-based hour of day field should be displ ayed when printing a time.
HOUR_OF_DAY1_FIELD	The value for the style used to indicate that the one-based hour of day field should be displ ayed when printing a time.
HOUR0_FIELD	The value for the style used to indicate that the zero-based hour field should be displayed w hen printing a time.
HOUR1_FIELD	The value for the style used to indicate that the one-based hour field should be displayed w hen printing a time.
LONG	The value for the style used to indicate that the long length style date and time should be pr inted.

MEDIUM	The value for the style used to indicate that the medium length style date and time should be printed.
MILLISECOND_FIELD	The value for the style used to indicate that the milliseconds field should be displayed when printing a time.
MINUTE_FIELD	The value for the style used to indicate that the minute field should be displayed when printing a time.
MONTH_FIELD	The value for the style used to indicate that the month field should be displayed when printing a date.
numberFormat	The NumberFormat object used to format and parse a date and a time.
SECOND_FIELD	The value for the style used to indicate that the seconds field should be displayed when printing a time.
SHORT	The value for the style used to indicate that the short length style date and time should be printed.
TIMEZONE_FIELD	The value for the style used to indicate that the time zone field should be displayed when printing a time.
WEEK_OF_MONTH_FIELD	The value for the style used to indicate that the week of the month field should be displayed when printing a date.
WEEK_OF_YEAR_FIELD	The value for the style used to indicate that the week of the year field should be displayed when printing a date.
YEAR_FIELD	The value for the style used to indicate that the year field should be displayed when printing a date.

Public Methods

Name	Description
equals	Overridden. Determines whether two DateFormat objects are equal.
format	Overloaded. Formats a Date object into a string.
getAvailableLocales	Returns an array of all the Locale objects that are available.
getCalendar	Gets the Calendar object that is being used. The calendar is used to determine the format of a date.
getDateInstance	Overloaded. Creates a new instance of a DateFormat object used to represent a date (without a time).
getDateTimeInstance	Overloaded. Creates a new instance of a DateFormat object used to represent a date and a time.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getInstance	Creates a new instance of a DateFormat object used to represent a date and a time with the default locale.
getNumberFormat	Gets the NumberFormat object used to format and parse a date and a time.

getTimeInstance	Overloaded. Creates a new instance of a DateFormat object used to represent a time (without a date).
getTimeZone	Gets the TimeZone object associated with the Calendar object of this instance.
getClass	(inherited from Object)
isLenient	Determines whether parsing is set to lenient for this DateFormat object. Leniency is determined by the C alendar object associated with this instance.
clone	Creates a new instance of a DateFormat object that is a shallow copy of this instance.
parse	Overloaded. Parses a string into a Date object.
parseObject	Overloaded. Parses a string into an Object.
setCalendar	Sets the Calendar object that is being used. The calendar is used to determine the format of a date.
setLenient	Sets a value that determines whether parsing is set to lenient for this DateFormat object. Leniency is dete rmined by the Calendar object associated with this instance.
setNumberFormat	Sets the NumberFormat object used to format and parse a date and a time.
setTimeZone	Sets a TimeZone object to be associated with the Calendar object of this instance.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DateFormat Class

Concepts

java.text Package

DateFormat Fields

Public Fields

Name	Description
AM_PM_FIELD	The value for the style used to indicate that the AM or PM field should be displayed when printing a time.
calendar	The underlying Calendar object being used for dates and times.
DATE_FIELD	The value for the style used to indicate that the date field should be displayed when printing a date.
DAY_OF_WEEK_FIELD	The value for the style used to indicate that the day of the week field should be displayed when printing a date.
DAY_OF_WEEK_IN_MONTH_FIELD	The value for the style used to indicate that the day of the week in the month field should be displayed when printing a date.
DAY_OF_YEAR_FIELD	The value for the style used to indicate that the day of the year field should be displayed when printing a date.
DEFAULT	The value for the style used to indicate that the default fields should be displayed when printing a date and time.
ERA_FIELD	The value for the style used to indicate that the era field should be displayed when printing a date.
FULL	The value for the style used to indicate that all fields should be displayed when printing a date.
HOUR_OF_DAY0_FIELD	The value for the style used to indicate that the zero-based hour of day field should be displayed when printing a time.
HOUR_OF_DAY1_FIELD	The value for the style used to indicate that the one-based hour of day field should be displayed when printing a time.
HOUR0_FIELD	The value for the style used to indicate that the zero-based hour field should be displayed when printing a time.
HOUR1_FIELD	The value for the style used to indicate that the one-based hour field should be displayed when printing a time.
LONG	The value for the style used to indicate that the long length style date and time should be printed.
MEDIUM	The value for the style used to indicate that the medium length style date and time should be printed.
MILLISECOND_FIELD	The value for the style used to indicate that the milliseconds field should be displayed when printing a time.
MINUTE_FIELD	The value for the style used to indicate that the minute field should be displayed when printing a time.

MONTH_FIELD	The value for the style used to indicate that the month field should be displayed when printing a date.
numberFormat	The NumberFormat object used to format and parse a date and a time.
SECOND_FIELD	The value for the style used to indicate that the seconds field should be displayed when printing a time.
SHORT	The value for the style used to indicate that the short length style date and time should be printed.
TIMEZONE_FIELD	The value for the style used to indicate that the time zone field should be displayed when printing a time.
WEEK_OF_MONTH_FIELD	The value for the style used to indicate that the week of the month field should be displayed when printing a date.
WEEK_OF_YEAR_FIELD	The value for the style used to indicate that the week of the year field should be displayed when printing a date.
YEAR_FIELD	The value for the style used to indicate that the year field should be displayed when printing a date.

See Also

Reference

DateFormat Class

Concepts

java.text Package

DateFormat.AM_PM_FIELD Field

The value for the style used to indicate that the AM or PM field should be displayed when printing a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int AM_PM_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.calendar Field

The underlying Calendar object being used for dates and times.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
protected java.util.Calendar calendar;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.DATE_FIELD Field

The value for the style used to indicate that the date field should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int DATE_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.DAY_OF_WEEK_FIELD Field

The value for the style used to indicate that the day of the week field should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int DAY_OF_WEEK_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.DAY_OF_WEEK_IN_MONTH_FIELD Field

The value for the style used to indicate that the day of the week in the month field should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int DAY_OF_WEEK_IN_MONTH_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.DAY_OF_YEAR_FIELD Field

The value for the style used to indicate that the day of the year field should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int DAY_OF_YEAR_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.DEFAULT Field

The value for the style used to indicate that the default fields should be displayed when printing a date and time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int DEFAULT;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.ERA_FIELD Field

The value for the style used to indicate that the era field should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int ERA_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.FULL Field

The value for the style used to indicate that all fields should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int FULL;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.HOUR_OF_DAY0_FIELD Field

The value for the style used to indicate that the zero-based hour of day field should be displayed when printing a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int HOUR_OF_DAY0_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.HOUR_OF_DAY1_FIELD Field

The value for the style used to indicate that the one-based hour of day field should be displayed when printing a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int HOUR_OF_DAY1_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.HOUR0_FIELD Field

The value for the style used to indicate that the zero-based hour field should be displayed when printing a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int HOUR0_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.HOUR1_FIELD Field

The value for the style used to indicate that the one-based hour field should be displayed when printing a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int HOUR1_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.LONG Field

The value for the style used to indicate that the long length style date and time should be printed.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int LONG;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.MEDIUM Field

The value for the style used to indicate that the medium length style date and time should be printed.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int MEDIUM;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.MILLISECOND_FIELD Field

The value for the style used to indicate that the milliseconds field should be displayed when printing a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int MILLISECOND_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.MINUTE_FIELD Field

The value for the style used to indicate that the minute field should be displayed when printing a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int MINUTE_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.MONTH_FIELD Field

The value for the style used to indicate that the month field should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int MONTH_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.numberFormat Field

The NumberFormat object used to format and parse a date and a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
protected java.text.NumberFormat numberFormat;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.SECOND_FIELD Field

The value for the style used to indicate that the seconds field should be displayed when printing a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int SECOND_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.SHORT Field

The value for the style used to indicate that the short length style date and time should be printed.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int SHORT;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.TIMEZONE_FIELD Field

The value for the style used to indicate that the time zone field should be displayed when printing a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int TIMEZONE_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.WEEK_OF_MONTH_FIELD Field

The value for the style used to indicate that the week of the month field should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int WEEK_OF_MONTH_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.WEEK_OF_YEAR_FIELD Field

The value for the style used to indicate that the week of the year field should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int WEEK_OF_YEAR_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.YEAR_FIELD Field

The value for the style used to indicate that the year field should be displayed when printing a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int YEAR_FIELD;
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat Constructor

Initializes a new instance of a DateFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
protected java.text.DateFormat();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat Methods

Public Methods

Name	Description
equals	Overridden. Determines whether two DateFormat objects are equal.
format	Overloaded. Formats a Date object into a string.
getAvailableLocales	Returns an array of all the Locale objects that are available.
getCalendar	Gets the Calendar object that is being used. The calendar is used to determine the format of a date.
getDateInstance	Overloaded. Creates a new instance of a DateFormat object used to represent a date (without a time).
getDateTimeInstance	Overloaded. Creates a new instance of a DateFormat object used to represent a date and a time.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and da ta structures like a hash table.
getInstance	Creates a new instance of a DateFormat object used to represent a date and a time with the default locale .
getNumberFormat	Gets the NumberFormat object used to format and parse a date and a time.
getTimeInstance	Overloaded. Creates a new instance of a DateFormat object used to represent a time (without a date).
getTimeZone	Gets the TimeZone object associated with the Calendar object of this instance.
getClass	(inherited from Object)
isLenient	Determines whether parsing is set to lenient for this DateFormat object. Leniency is determined by the C alendar object associated with this instance.
clone	Creates a new instance of a DateFormat object that is a shallow copy of this instance.
parse	Overloaded. Parses a string into a Date object.
parseObject	Overloaded. Parses a string into an Object.
setCalendar	Sets the Calendar object that is being used. The calendar is used to determine the format of a date.
setLenient	Sets a value that determines whether parsing is set to lenient for this DateFormat object. Leniency is dete rmined by the Calendar object associated with this instance.
setNumberFormat	Sets the NumberFormat object used to format and parse a date and a time.
setTimeZone	Sets a TimeZone object to be associated with the Calendar object of this instance.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description

finalize	(inherited from Object)

See Also

Reference

DateFormat Class

Concepts

java.text Package

DateFormat.clone Method

Creates a new instance of a DateFormat object that is a shallow copy of this instance.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object clone();
```

Return Value

A new instance of a DateFormat object that is a shallow copy of this instance.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.equals Method

Determines whether two DateFormat objects are equal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

A DateFormat object to compare with the current DateFormat object for equality.

Return Value

true if the two DateFormat objects are equal; false otherwise.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.format Method

Formats a Date object into a string.

Overload List

Name	Description
DateFormat.format (Date)	Formats a Date object into a string.
DateFormat.format (Object)	Formats an object into a string.
DateFormat.format (Date, StringBuffer, FieldPosition)	Formats a Date object and appends it to a StringBuffer object.
DateFormat.format (Object, StringBuffer, FieldPosition)	Formats an object representing a Date and appends it to a StringBuffer object.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.format Method (Date)

Formats a Date object into a string.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String format(
    java.util.Date date);
```

Parameters

date

The Date object to be formatted into a string.

Return Value

A String object containing the formatted Date object.

Example

The following example shows how to format a Date object.

```
// dateformat_format.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Apply a pattern to a SimpleDateFormat object.
        String pattern =
            "'This moment: 'dd MMM yyyy '@'hh 'hours and 'mm 'minutes.'";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Print out the time now using the format method.
        System.out.println(sdf.format(new Date()));
    }
}

/*
Sample Output:
This moment: 11 Nov 2004 @01 hours and 14 minutes.
*/
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.format Method (Date, StringBuffer, FieldPosition)

Formats a Date object and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.StringBuffer format(
    java.util.Date date,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition pos);
```

Parameters

date

The Date object to be formatted into a string.

appendBuf

A StringBuffer object to be appended with the Date object formatted into a string.

pos

Used to find start and end indices of a particular field of the date in the formatted string.

Return Value

A StringBuffer object appended with the formatted Date object.

Example

The following example shows how to format a Date object and how to determine the indices of the year field.

```
// dateformat_format1.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Apply a pattern to a SimpleDateFormat object.
        String pattern =
            "'This moment: 'dd MMM yyyy '@'hh 'hours and 'mm 'minutes.'";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Print out the time now using the format method.
        StringBuffer buf = new StringBuffer();
        FieldPosition pos = new FieldPosition(DateFormat.YEAR_FIELD);
        sdf.format(new Date(), buf, pos);
        System.out.println(buf);
        System.out.println("The year field occupies indices: " +
            pos.getBeginIndex() + " - " + pos.getEndIndex());
    }
}

/*
Sample Output:
This moment: 11 Nov 2004 @10 hours and 29 minutes.
The year field occupies indices: 20 - 24
*/
```

See Also

Reference

DateFormat Class
Concepts
DateFormat Members
java.text Package

DateFormat.format Method (Object, StringBuffer, FieldPosition)

Formats an object representing a Date and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.StringBuffer format(
    java.lang.Object obj,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition pos);
```

Parameters

obj

The object representing a date to be formatted into a string.

appendBuf

A StringBuffer object to be appended with the object formatted into a string.

pos

Used to find start and end indices of a particular field of the date in the formatted string.

Return Value

A StringBuffer object appended with the formatted object representing a date.

Example

The following example shows how to format a Date object and how to determine the indices of the year field.

```
// dateformat_format2.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Apply a pattern to a SimpleDateFormat object.
        String pattern =
            "'This moment: 'dd MMM yyyy '@'hh 'hours and 'mm 'minutes.'";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Print out the time now using the format method.
        StringBuffer buf = new StringBuffer();
        FieldPosition pos = new FieldPosition(DateFormat.YEAR_FIELD);
        sdf.format(new Date(), buf, pos);
        System.out.println(buf);
        System.out.println("The year field occupies indices: " +
            pos.getBeginIndex() + " - " + pos.getEndIndex());
    }
}

/*
Sample Output:
This moment: 11 Nov 2004 @10 hours and 29 minutes.
The year field occupies indices: 20 - 24
*/
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getAvailableLocales Method

Returns an array of all the Locale objects that are available.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Locale[] getAvailableLocales();
```

Return Value

An array of all the Locale objects that are available.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getCalendar Method

Gets the Calendar object that is being used. The calendar is used to determine the format of a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.util.Calendar getCalendar();
```

Return Value

The Calendar object that is being used.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getDateInstance Method

Creates a new instance of a DateFormat object used to represent a date (without a time).

Overload List

Name	Description
DateFormat.getDateInstance ()	Creates a new instance of a DateFormat object used to represent a date (without a time) with the default style and locale.
DateFormat.getDateInstance (int)	Creates a new instance of a DateFormat object used to represent a date (without a time) with the given style and the default locale.
DateFormat.getDateInstance (int, Locale)	Creates a new instance of a DateFormat object used to represent a date (without a time) with the given style and locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getDateInstance Method ()

Creates a new instance of a DateFormat object used to represent a date (without a time) with the default style and locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getDateInstance();
```

Return Value

A new instance of a DateFormat object used to represent a date (without a time) with the default style and the default locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getDateInstance Method (Int32)

Creates a new instance of a DateFormat object used to represent a date (without a time) with the given style and the default locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getDateInstance(
    int style);
```

Parameters

style

The style used when formatting the date. This can be one of the four style constants defined by DateFormat class -- SHORT, MEDIUM, LONG, and FULL.

Return Value

A new instance of a DateFormat object used to represent a date (without a time) with the given style and the default locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getDateInstance Method (Int32, Locale)

Creates a new instance of a DateFormat object used to represent a date (without a time) with the given style and locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getDateInstance(
    int style,
    java.util.Locale loc);
```

Parameters

style

The style used when formatting the date. This can be one of the four style constants defined by DateFormat class -- SHORT, MEDIUM, LONG, and FULL.

loc

A Locale object representing the locale used when formatting the date.

Return Value

A new instance of a DateFormat object used to represent a date (without a time) with the given style and locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getDateTimeInstance Method

Creates a new instance of a DateFormat object used to represent a date and a time.

Overload List

Name	Description
DateFormat.getDateTimeInstance ()	Creates a new instance of a DateFormat object used to represent a date and a time with the default style and locale.
DateFormat.getDateTimeInstance (int, int)	Creates a new instance of a DateFormat object used to represent a date and a time with the given styles and the default locale.
DateFormat.getDateTimeInstance (int, int, Locale)	Creates a new instance of a DateFormat object used to represent a date and a time with the given styles and locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getDateTimeInstance Method ()

Creates a new instance of a DateFormat object used to represent a date and a time with the default style and locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getDateTimeInstance();
```

Return Value

A new instance of a DateFormat object used to represent a date and a time with the default style and locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getDateTimeInstance Method (Int32, Int32)

Creates a new instance of a DateFormat object used to represent a date and a time with the given styles and the default locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getDateTimeInstance(
    int dateStyle,
    int timeStyle);
```

Parameters

dateStyle

The style used when formatting the date. This can be one of the four style constants defined by DateFormat class -- SHORT, MEDIUM, LONG, and FULL.

timeStyle

The style used when formatting the time. This can be one of the four style constants defined by DateFormat class -- SHORT, MEDIUM, LONG, and FULL.

Return Value

A new instance of a DateFormat object used to represent a date and a time with the given styles and the default locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getDateTimeInstance Method (Int32, Int32, Locale)

Creates a new instance of a DateFormat object used to represent a date and a time with the given styles and locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getDateTimeInstance(
    int dateStyle,
    int timeStyle,
    java.util.Locale loc);
```

Parameters

dateStyle

The style used when formatting the date. This can be one of the four style constants defined by DateFormat class -- SHORT, MEDIUM, LONG, and FULL.

timeStyle

The style used when formatting the time. This can be one of the four style constants defined by DateFormat class -- SHORT, MEDIUM, LONG, and FULL.

loc

A Locale object representing the locale used when formatting the date and time.

Return Value

A new instance of a DateFormat object used to represent a date and a time with the given styles and locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.hashCode Method

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

A hash code for the current object.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getInstance Method

Creates a new instance of a DateFormat object used to represent a date and a time with the default locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getInstance();
```

Return Value

A new instance of a DateFormat object used to represent a date and a time with the default locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getNumberFormat Method

Gets the NumberFormat object used by this formatter to format and/or parse various components of dates and times.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.NumberFormat getNumberFormat();
```

Return Value

The NumberFormat object used to format and parse a date and a time.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getTimeInstance Method

Creates a new instance of a DateFormat object used to represent a time (without a date).

Overload List

Name	Description
DateFormat.getTimeInstance ()	Creates a new instance of a DateFormat object used to represent a time (without a date) with the default style and locale.
DateFormat.getTimeInstance (int)	Creates a new instance of a DateFormat object used to represent a time (without a date) with the given style and the default locale.
DateFormat.getTimeInstance (int, Locale)	Creates a new instance of a DateFormat object used to represent a time (without a date) with the given style and locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getTimeInstance Method ()

Creates a new instance of a DateFormat object used to represent a time (without a date) with the default style and locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getTimeInstance();
```

Return Value

A new instance of a DateFormat object used to represent a time (without a date) with the default style and locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getTimeInstance Method (Int32)

Creates a new instance of a DateFormat object used to represent a time (without a date) with the given style and the default locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getTimeInstance(
    int style);
```

Parameters

style

The style used when formatting the time. This can be one of the four style constants defined by DateFormat class -- SHORT, MEDIUM, LONG, and FULL.

Return Value

A new instance of a DateFormat object used to represent a time (without a date) with the given style and the default locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getTimeInstance Method (Int32, Locale)

Creates a new instance of a DateFormat object used to represent a time (without a date) with the given style and locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.DateFormat getTimeInstance(
    int style,
    java.util.Locale loc);
```

Parameters

style

The style used when formatting the time. This can be one of the four style constants defined by DateFormat class -- SHORT, MEDIUM, LONG, and FULL.

loc

A Locale object representing the locale used when formatting the time.

Return Value

A new instance of a DateFormat object used to represent a time (without a date) with the given style and locale.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.getTimeZone Method

Gets the TimeZone object associated with the Calendar object of this instance.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.util.TimeZone getTimeZone();
```

Return Value

The TimeZone object associated with the Calendar object of this instance.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.isLenient Method

Determines whether parsing is set to lenient for this DateFormat object. Leniency is determined by the Calendar object associated with this instance.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean isLenient();
```

Return Value

true if parsing is set to lenient for this DateFormat object; false otherwise.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.parse Method

Parses a string into a Date object.

Overload List

Name	Description
DateFormat.parse (String)	Parses a string into a Date object.
DateFormat.parse (String, ParsePosition)	Parses a string into a Date object.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.parse Method (String)

Parses a string into a Date object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.util.Date parse(
    java.lang.String sourceStr) throws java.text.ParseException;
```

Parameters

sourceStr

The string to be parsed into a Date object.

Return Value

A Date object containing the date represented by the string.

Example

The following example shows how to parse a string into a formatted date.

```
// dateformat_parse.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a SimpleDateFormat object.
        String pattern = "dd-MMM-yyyy HH:mm:ss.SSS";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Parse a sample date starting at position 0.
        String sampleDateStr = "04-Jul-1776 22:00:00.000";
        Date sampleDate = null;
        try
        {
            sampleDate = sdf.parse(sampleDateStr);
        }
        catch (ParseException ex)
        {
            System.out.println(ex.toString());
        }

        // Print out the date using the supplied pattern.
        System.out.println(sdf.format(sampleDate));
    }
}

/*
Sample Output:
04-Jul-1776 22:00:00.000
*/
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.parse Method (String, ParsePosition)

Parses a string into a Date object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Date parse(
    java.lang.String sourceStr,
    java.text.ParsePosition pos);
```

Parameters

sourceStr

The string to be parsed into a Date object.

pos

Used to indicate the first position in the string to be parsed.

Return Value

A Date object containing the date represented by the string.

Example

The following example shows how to parse a string into a formatted date.

```
// dateformat_parse1.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a SimpleDateFormat object.
        String pattern = "dd-MMM-yyyy HH:mm:ss.SSS";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Parse a sample date starting at position 0.
        String sampleDateStr = "04-Jul-1776 22:00:00.000";
        ParsePosition pos = new ParsePosition(0);
        Date sampleDate = sdf.parse(sampleDateStr, pos);

        // Print out the date using the supplied pattern.
        System.out.println(sdf.format(sampleDate));
    }
}

/*
Sample Output:
04-Jul-1776 22:00:00.000
*/
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.parseObject Method

Parses a string into an Object.

Overload List

Name	Description
DateFormat.parseObject (String)	Parses a string into an Object.
DateFormat.parseObject (String, ParsePosition)	Parses a string into an Object.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.parseObject Method (String, ParsePosition)

Parses a string into an Object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object parseObject(
    java.lang.String sourceStr,
    java.text.ParsePosition pos);
```

Parameters

sourceStr

The string to be parsed into an Object.

pos

Used to indicate the first position in the string to be parsed.

Return Value

An Object containing the date represented by the string.

Example

The following example shows how to parse a string into a formatted date.

```
// dateformat_parseobject.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a SimpleDateFormat object.
        String pattern = "dd-MMM-yyyy HH:mm:ss.SSS";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Parse a sample date starting at position 0.
        String sampleDateStr = "04-Jul-1776 22:00:00.000";
        ParsePosition pos = new ParsePosition(0);
        Date sampleDate = (Date)sdf.parseObject(sampleDateStr, pos);

        // Print out the date using the supplied pattern.
        System.out.println(sdf.format(sampleDate));
    }
}

/*
Sample Output:
04-Jul-1776 22:00:00.000
*/
```

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.setCalendar Method

Sets the Calendar object that is being used. The calendar is used to determine the format of a date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setCalendar(
    java.util.Calendar newCalendar);
```

Parameters

newCalendar

The Calendar object that is being used.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.setLenient Method

Sets a value that determines whether parsing is set to lenient for this DateFormat object. Leniency is determined by the Calendar object associated with this instance.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setLenient(
    boolean lenient);
```

Parameters

lenient

A value that determines whether parsing is set to lenient for this DateFormat object.

See Also
Reference
DateFormat Class
Concepts
DateFormat Members
java.text Package

DateFormat.setNumberFormat Method

Sets the NumberFormat object used to format and parse a date and a time.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setNumberFormat(
    java.text.NumberFormat numFormat);
```

Parameters

numFormat

The NumberFormat object used to format and parse a date and a time.

See Also

Reference

DateFormat Class

Concepts

DateFormat Members

java.text Package

DateFormat.setTimeZone Method

Sets a TimeZone object to be associated with the Calendar object of this instance.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setTimeZone(
    java.util.TimeZone zone);
```

Parameters

zone

A TimeZone object to be associated with the Calendar object of this instance.

See Also
Reference
DateFormat Class
Concepts
DateFormat Members
java.text Package

DateFormatSymbols Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public class java.text.DateFormatSymbols
    extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable
```

Inheritance Hierarchy

java.lang.Object

java.text.DateFormatSymbols

See Also

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols Members

The following tables list the members exposed by the DateFormatSymbols type.

Public Constructors

Name	Description
DateFormatSymbols	Overloaded.

Public Methods

Name	Description
equals	Overridden.
getAmPmStrings	
getEras	
hashCode	Overridden.
getLocalPatternChars	
getMonths	
getShortMonths	
getShortWeekdays	
getClass	(inherited from Object)
getWeekdays	
getZoneStrings	
MemberwiseClone	Performs a shallow copy of the members.
setAmPmStrings	
setEras	
setLocalPatternChars	
setMonths	
setShortMonths	
setShortWeekdays	
setWeekdays	
setZoneStrings	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DateFormatSymbols Class

Concepts

java.text Package

DateFormatSymbols Constructor

Overload List

Name	Description
DateFormatSymbols ()	
DateFormatSymbols (Locale)	

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols Constructor ()

Initializes a new instance of the DateFormatSymbols Class .

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.DateFormatSymbols();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols Constructor (Locale)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.DateFormatSymbols(
    java.util.Locale loc);
```

Parameters

loc

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols Methods

Public Methods

Name	Description
equals	Overridden.
getAmPmStrings	
getEras	
hashCode	Overridden.
getLocalPatternChars	
getMonths	
getShortMonths	
getShortWeekdays	
getClass	(inherited from Object)
getWeekdays	
getZoneStrings	
MemberwiseClone	Performs a shallow copy of the members.
setAmPmStrings	
setEras	
setLocalPatternChars	
setMonths	
setShortMonths	
setShortWeekdays	
setWeekdays	
setZoneStrings	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference
DateFormatSymbols Class
Concepts
java.text Package

DateFormatSymbols.clone Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.equals Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.getAmPmStrings Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String[] getAmPmStrings();
```

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.getEras Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String[] getEras();
```

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.hashCode Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.getLocalPatternChars Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getLocalPatternChars();
```

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.getMonths Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String[] getMonths();
```

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.getShortMonths Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String[] getShortMonths();
```

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.getShortWeekdays Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String[] getShortWeekdays();
```

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.getWeekdays Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String[] getWeekdays();
```

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.getZoneStrings Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public [Ljava.lang.String;[] getZoneStrings();
```

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DateFormatSymbols.setAmPmStrings Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setAmPmStrings(
    java.lang.String[] newAmPms);
```

Parameters

newAmPms

See Also
Reference
DateFormatSymbols Class
Concepts
DateFormatSymbols Members
java.text Package

DateFormatSymbols.setEras Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setEras(
    java.lang.String[] newEras);
```

Parameters
newEras

See Also
Reference
DateFormatSymbols Class
Concepts
DateFormatSymbols Members
java.text Package

DateFormatSymbols.setLocalPatternChars Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setLocalPatternChars(
    java.lang.String newLocalPatternChars);
```

Parameters

newLocalPatternChars

See Also
Reference
DateFormatSymbols Class
Concepts
DateFormatSymbols Members
java.text Package

DateFormatSymbols.setMonths Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setMonths(
    java.lang.String[] newMonths);
```

Parameters
newMonths

See Also
Reference
DateFormatSymbols Class
Concepts
DateFormatSymbols Members
java.text Package

DateFormatSymbols.setShortMonths Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setShortMonths(
    java.lang.String[] newShortMonths);
```

Parameters
newShortMonths

See Also
Reference
DateFormatSymbols Class
Concepts
DateFormatSymbols Members
java.text Package

DateFormatSymbols.setShortWeekdays Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setShortWeekdays(
    java.lang.String[] newShortWeekdays);
```

Parameters

newShortWeekdays

See Also
Reference
DateFormatSymbols Class
Concepts
DateFormatSymbols Members
java.text Package

DateFormatSymbols.setWeekdays Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setWeekdays(
    java.lang.String[] newWeekdays);
```

Parameters
newWeekdays

See Also
Reference
DateFormatSymbols Class
Concepts
DateFormatSymbols Members
java.text Package

DateFormatSymbols.setZoneStrings Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setZoneStrings(
    [Ljava.lang.String;[] newZoneStrings);
```

Parameters

newZoneStrings

See Also

Reference

DateFormatSymbols Class

Concepts

DateFormatSymbols Members

java.text Package

DecimalFormat Class

Contains methods and properties used to format a number as a string and to parse a string into a number. This class provides a default implementation of the abstract class NumberFormat.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public class java.text.DecimalFormat
    extends java.text.NumberFormat
```

Example

The following example demonstrates the applyPattern,format, and parse methods of the DecimalFormat class.

```
// decimalformat_overview.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a DecimalFormat object and set it's pattern.
        DecimalFormat decForm = new DecimalFormat();
        decForm.applyPattern("#.00###");

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos1 =
            new FieldPosition(NumberFormat.FRACTION_FIELD);
        FieldPosition pos2 =
            new FieldPosition(NumberFormat.INTEGER_FIELD);

        dest1 = decForm.format(22.3423D, dest1, pos1);
        System.out.println("dest1 = " + dest1);
        System.out.println("FRACTION is at: " + pos1.getBeginIndex() +
            ", " + pos1.getEndIndex());

        dest2 = decForm.format(64000D, dest2, pos2);
        System.out.println("dest2 = " + dest2);
        System.out.println("INTEGER portion is at: " +
            pos2.getBeginIndex() + ", " + pos2.getEndIndex());

        // Parse the decimal starting at position 8.
        ParsePosition pos3 = new ParsePosition(8);
        Number n = decForm.parse("123456789.14", pos3);
        System.out.println("Number = " + n);
    }
}

/*
Output:
dest1 = 22.3423
FRACTION is at: 3, 7
dest2 = 64000.00
INTEGER portion is at: 0, 5
Number = 9.14
*/
```

Inheritance Hierarchy

java.lang.Object

  java.text.Format

    java.text.NumberFormat

      java.text.DecimalFormat

See Also

Concepts

DecimalFormat Members

java.text Package

DecimalFormat Members

Contains methods and properties used to format a number as a string and to parse a string into a number. This class provides a default implementation of the abstract class NumberFormat.

The following tables list the members exposed by the DecimalFormat type.

Public Constructors

Name	Description
DecimalFormat	Overloaded. Initializes a new instance of a DecimalFormat object.

Public Methods

Name	Description
applyLocalizedPattern	Changes the pattern that an instance of a DecimalFormat object is using after it is created. Localization of the pattern is currently not supported by the J# Class Libraries.
applyPattern	Changes the pattern that an instance of a DecimalFormat object is using after it is created.
equals	Overridden. Determines whether two DecimalFormat objects are equal.
format	Overloaded. Formats a decimal into a string.
getDecimalFormatSymbols	Gets the DecimalFormatSymbols object that contains the symbols used when formatting a decimal.
getGroupingSize	Gets the number of characters that are grouped together between grouping separators.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algo rithms and data structures like a hash table.
getMaximumFractionDigits	Gets a value representing the maximum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
getMaximumIntegerDigits	Gets a value representing the maximum number of digits displayed in the integer part of t he number. (inherited from NumberFormat)
getMinimumFractionDigits	Gets a value representing the minimum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
getMinimumIntegerDigits	Gets a value representing the minimum number of digits displayed in the integer part of th e number. (inherited from NumberFormat)
getMultiplier	Gets the multiplier used for calculating percents or permills.
getNegativePrefix	Gets the symbol that is prefixed to a negative number.
getNegativeSuffix	Gets the symbol that is appended to a negative number.
getPositivePrefix	Gets the symbol that is prefixed to a positive number.
getPositiveSuffix	Gets the symbol that is appended to a positive number.
getClass	(inherited from Object)

isDecimalSeparatorAlwaysShown	Determines whether the decimal separator is shown with integers.
isGroupingUsed	Determines whether digits are being grouped together. (inherited from NumberFormat)
isParseIntegerOnly	Determines whether only the integer portion of the number is being parsed or if the fractional part is also included. (inherited from NumberFormat)
clone	Creates a shallow copy of the current DecimalFormat object.
parse	Overloaded. Parses a string into a decimal.
parseObject	Overloaded. Parses a string into an Object. (inherited from NumberFormat)
setDecimalFormatSymbols	Sets a DecimalFormatSymbols object that contains the symbols used when formatting a decimal.
setDecimalSeparatorAlwaysShown	Sets a value indicating whether the decimal separator is shown with integers.
setGroupingSize	Sets the number of characters that are grouped together between grouping separators.
setGroupingUsed	Sets a value determining whether digits are being grouped together. (inherited from NumberFormat)
setMaximumFractionDigits	Sets the value representing the maximum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
setMaximumIntegerDigits	Sets the value representing the maximum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
setMinimumFractionDigits	Sets the value representing the minimum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
setMinimumIntegerDigits	Sets the value representing the minimum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
setMultiplier	Sets the multiplier used for calculating percents or permills.
setNegativePrefix	Sets the symbol that is prefixed to a negative number.
setNegativeSuffix	Sets the symbol that is appended to a negative number.
setParseIntegerOnly	Sets a value determining whether only the integer portion of the number is being parsed or if the fractional part is also included. (inherited from NumberFormat)
setPositivePrefix	Sets the symbol that is prefixed to a positive number.
setPositiveSuffix	Sets the symbol that is appended to a positive number.
toLocalizedPattern	Applies the previously set pattern to the string for an instance of a DecimalFormat object. Localization of the pattern is currently not supported by the J# Class Libraries.
toPattern	Applies the previously set pattern to the string for an instance of a DecimalFormat object.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DecimalFormat Class

Concepts

java.text Package

DecimalFormat Constructor

Initializes a new instance of a DecimalFormat object.

Overload List

Name	Description
DecimalFormat ()	Initializes a new instance of a DecimalFormat object.
DecimalFormat (String)	Initializes a new instance of a DecimalFormat object. The decimal will be formatted into a string using the pattern provided.
DecimalFormat (String, DecimalFormatSymbols)	Initializes a new instance of a DecimalFormat object. The decimal will be formatted into a string using the pattern and format symbols provided.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat Constructor ()

Initializes a new instance of a DecimalFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.DecimalFormat();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat Constructor (String)

Initializes a new instance of a DecimalFormat object. The decimal will be formatted into a string using the pattern provided.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.DecimalFormat(
    java.lang.String pattern);
```

Parameters

pattern

The pattern describing how to format the decimal as a string.

Remarks

Valid characters for the pattern string include "#", "0", ".", ",", "%", ";", and "\u2030" (the permill operator). As positive and negative numbers may have a different format, two different patters can be specified by this pattern string with a semicolon separator in between the two patterns. Following are short descriptions for each of the pattern characters:

Character	Definition
#	Place holder for a digit. Ignored if the number being formatted does not contain a digit to place at this position.
0	Place holder for a digit. If the number being formatted does not contain a digit to place at this position, a zero is placed there.
.	Place holder for a decimal separator.
,	Place holder for a grouping separator.
%	Indicator for a percentage pattern. If present, the formatter will divide the number by 100 and show the percent symbol.
\u2030	Indicator for a permill pattern. If present, the formatter will divide the number by 1000 and show the permill symbol.
;	Separates positive and negative format patterns.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat Constructor (String, DecimalFormatSymbols)

Initializes a new instance of a DecimalFormat object. The decimal will be formatted into a string using the pattern and format symbols provided.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.DecimalFormat(
    java.lang.String pattern,
    java.text.DecimalFormatSymbols symbols);
```

Parameters

pattern

The pattern describing how to format the decimal as a string.

symbols

A DecimalFormatSymbols object describing the symbols to use in formatting the decimal as a string.

Remarks

Valid characters for the pattern string include "#", "0", ".", ",", "%", ";", and "\u2030" (the permill operator). As positive and negative numbers may have a different format, two different patters can be specified by this pattern string with a semicolon separator in between the two patterns. Following are short descriptions for each of the pattern characters:

Character	Definition
#	Place holder for a digit. Ignored if the number being formatted does not contain a digit to place at this position.
0	Place holder for a digit. If the number being formatted does not contain a digit to place at this position, a zero is placed there.
.	Place holder for a decimal separator.
,	Place holder for a grouping separator.
%	Indicator for a percentage pattern. If present, the formatter will divide the number by 100 and show the percent symbol.
\u2030	Indicator for a permill pattern. If present, the formatter will divide the number by 1000 and show the permill symbol.
;	Separates positive and negative format patterns.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat Methods

Public Methods

Name	Description
applyLocalizedPattern	Changes the pattern that an instance of a DecimalFormat object is using after it is created. Localization of the pattern is currently not supported by the J# Class Libraries.
applyPattern	Changes the pattern that an instance of a DecimalFormat object is using after it is created.
equals	Overridden. Determines whether two DecimalFormat objects are equal.
format	Overloaded. Formats a decimal into a string.
getDecimalFormatSymbols	Gets the DecimalFormatSymbols object that contains the symbols used when formatting a decimal.
getGroupingSize	Gets the number of characters that are grouped together between grouping separators.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getMaximumFractionDigits	Gets a value representing the maximum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
getMaximumIntegerDigits	Gets a value representing the maximum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
getMinimumFractionDigits	Gets a value representing the minimum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
getMinimumIntegerDigits	Gets a value representing the minimum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
getMultiplier	Gets the multiplier used for calculating percents or permills.
getNegativePrefix	Gets the symbol that is prefixed to a negative number.
getNegativeSuffix	Gets the symbol that is appended to a negative number.
getPositivePrefix	Gets the symbol that is prefixed to a positive number.
getPositiveSuffix	Gets the symbol that is appended to a positive number.
getClass	(inherited from Object)
isDecimalSeparatorAlwaysShown	Determines whether the decimal separator is shown with integers.
isGroupingUsed	Determines whether digits are being grouped together. (inherited from NumberFormat)
isParseIntegerOnly	Determines whether only the integer portion of the number is being parsed or if the fractional part is also included. (inherited from NumberFormat)

clone	Creates a shallow copy of the current DecimalFormat object.
parse	Overloaded. Parses a string into a decimal.
parseObject	Overloaded. Parses a string into an Object. (inherited from NumberFormat)
setDecimalFormatSymbols	Sets a DecimalFormatSymbols object that contains the symbols used when formatting a de cimal.
setDecimalSeparatorAlwaysShown	Sets a value indicating whether the decimal separator is shown with integers.
setGroupingSize	Sets the number of characters that are grouped together between grouping separators.
setGroupingUsed	Sets a value determining whether digits are being grouped together. (inherited from NumberFormat)
setMaximumFractionDigits	Sets the value representing the maximum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
setMaximumIntegerDigits	Sets the value representing the maximum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
setMinimumFractionDigits	Sets the value representing the minimum number of digits displayed in the fractional part of the number. (inherited from NumberFormat)
setMinimumIntegerDigits	Sets the value representing the minimum number of digits displayed in the integer part of the number. (inherited from NumberFormat)
setMultiplier	Sets the multiplier used for calculating percents or permills.
setNegativePrefix	Sets the symbol that is prefixed to a negative number.
setNegativeSuffix	Sets the symbol that is appended to a negative number.
setParseIntegerOnly	Sets a value determining whether only the integer portion of the number is being parsed o r if the fractional part is also included. (inherited from NumberFormat)
setPositivePrefix	Sets the symbol that is prefixed to a positive number.
setPositiveSuffix	Sets the symbol that is appended to a positive number.
toLocalizedPattern	Applies the previously set pattern to the string for an instance of a DecimalFormat object. L ocalization of the pattern is currently not supported by the J# Class Libraries.
toPattern	Applies the previously set pattern to the string for an instance of a DecimalFormat object.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

DecimalFormat Class
Concepts
java.text Package

DecimalFormat.applyLocalizedPattern Method

Changes the pattern that an instance of a DecimalFormat object is using after it is created. Localization of the pattern is currently not supported by the J# Class Libraries.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void applyLocalizedPattern(
    java.lang.String pattern);
```

Parameters

pattern

Specifies the style used to format to or parse from number strings.

Example

The following example shows how to format two decimals using a localized pattern.

```
// decimalformat_applylocalizedpattern.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a DecimalFormat object and set it's pattern.
        DecimalFormat decForm = new DecimalFormat();
        decForm.applyLocalizedPattern("#.00###");

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.FRACTION_FIELD);

        dest1 = decForm.format(22.3423D, dest1, pos);
        System.out.println("dest1 = " + dest1);

        dest2 = decForm.format(64000D, dest2, pos);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 22.3423
dest2 = 64000.00
*/
```

Remarks

Localization of the pattern is currently not supported by the J# Class Libraries. Calling this method has the same effect as calling applyPattern.

Valid characters for the pattern string include "#", "0", ".", ",", "%", ";", and "\u2030" (the permill operator). As positive and negative numbers may have a different format, two different patters can be specified by this pattern string with a semicolon separator in between the two patterns. Following are short descriptions for each of the pattern characters:

Character	Definition

#	Place holder for a digit. Ignored if the number being formatted does not contain a digit to place at this position.
0	Place holder for a digit. If the number being formatted does not contain a digit to place at this position, a zero is place d there.
.	Place holder for a decimal separator.
,	Place holder for a grouping separator.
%	Indicator for a percentage pattern. If present, the formatter will divide the number by 100 and show the percent sym bol.
\u2030	Indicator for a permill pattern. If present, the formatter will divide the number by 1000 and show the permill symbol.
;	Separates positive and negative format patterns.

See Also
Reference
DecimalFormat Class
Concepts
DecimalFormat Members
java.text Package

DecimalFormat.applyPattern Method

Changes the pattern that an instance of a DecimalFormat object is using after it is created.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void applyPattern(
    java.lang.String pattern);
```

Parameters

pattern

Specifies the style used to format to or parse from number strings.

Example

The following example shows how to format two decimals using a localized pattern.

```
// decimalformat_applypattern.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a DecimalFormat object and set it's pattern.
        DecimalFormat decForm = new DecimalFormat();
        decForm.applyPattern("#.00###");

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.FRACTION_FIELD);

        dest1 = decForm.format(22.3423D, dest1, pos);
        System.out.println("dest1 = " + dest1);

        dest2 = decForm.format(64000D, dest2, pos);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 22.3423
dest2 = 64000.00
*/
```

Remarks

Valid characters for the pattern string include "#", "0", ".", ",", "%", ";", and "\u2030" (the permill operator). As positive and negative numbers may have a different format, two different patters can be specified by this pattern string with a semicolon separator in between the two patterns. Following are short descriptions for each of the pattern characters:

Character	Definition
#	Place holder for a digit. Ignored if the number being formatted does not contain a digit to place at this position.

0	Place holder for a digit. If the number being formatted does not contain a digit to place at this position, a zero is placed there.
.	Place holder for a decimal separator.
,	Place holder for a grouping separator.
%	Indicator for a percentage pattern. If present, the formatter will divide the number by 100 and show the percent symbol.
\u2030	Indicator for a permill pattern. If present, the formatter will divide the number by 1000 and show the permill symbol.
;	Separates positive and negative format patterns.

See Also
Reference
DecimalFormat Class
Concepts
DecimalFormat Members
java.text Package

DecimalFormat.clone Method

Creates a shallow copy of the current DecimalFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object clone();
```

Return Value

A shallow copy of the current DecimalFormat object.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.equals Method

Determines whether two DecimalFormat objects are equal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

A DecimalFormat object to compare with the current DecimalFormat object for equality.

Return Value

true if the two DecimalFormat objects are equal; false otherwise.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.format Method

Formats a decimal into a string.

Overload List

Name	Description
DecimalFormat.format (double)	Formats a double into a string.
DecimalFormat.format (long)	Formats a long into a string.
DecimalFormat.format (Object)	Formats an object into a string.
DecimalFormat.format (double, StringBuffer, FieldPosition)	Formats a double and appends it to a StringBuffer object.
DecimalFormat.format (long, StringBuffer, FieldPosition)	Formats a long and appends it to a StringBuffer object.
DecimalFormat.format (Object, StringBuffer, FieldPosition)	Formats an Object and appends it to a StringBuffer object.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.format Method (Double, StringBuffer, FieldPosition)

Formats a double and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer format(
    double dblnum,
    java.lang.StringBuffer dest,
    java.text.FieldPosition pos);
```

Parameters

dblnum

The double to be formatted into a string.

dest

A StringBuffer object to be appended with the double formatted into a string.

pos

Used to find start and end indices of a particular field of the number in the formatted string.

Return Value

A StringBuffer object appended with the double formatted into a string.

Example

The following example formats a decimal value and prints out the beginning and ending indices of the fraction part of the decimal.

```
// decimalformat_format.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a DecimalFormat object and set it's pattern.
        DecimalFormat decForm = new DecimalFormat();
        decForm.applyPattern("#.00###");

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.FRACTION_FIELD);

        dest1 = decForm.format(22.3423D, dest1, pos);
        System.out.println("dest1 = " + dest1);
        System.out.println("FRACTION is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());

        dest2 = decForm.format(64000D, dest2, pos);
        System.out.println("dest2 = " + dest2);
        System.out.println("FRACTION is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());
    }
}
```

```
/*
Output:
dest1 = 22.3423
FRACTION is at: 3, 7
dest2 = 64000.00
FRACTION is at: 6, 8
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.format Method (Int64, StringBuffer, FieldPosition)

Formats a long and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer format(
    long number,
    java.lang.StringBuffer dest,
    java.text.FieldPosition pos);
```

Parameters

number

The long to be formatted into a string.

dest

A StringBuffer object to be appended with the long formatted into a string.

pos

Used to find start and end indices of a particular field of the number in the formatted string.

Return Value

A StringBuffer object appended with the long formatted into a string.

Example

The following example formats a long value and prints out the beginning and ending indices of the integer part of the long.

```
// decimalformat_format_2.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a DecimalFormat object and set it's pattern.
        DecimalFormat decForm = new DecimalFormat();
        decForm.applyPattern("#.00###");

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.INTEGER_FIELD);

        dest1 = decForm.format(223423L, dest1, pos);
        System.out.println("dest1 = " + dest1);
        System.out.println("INTEGER fields are at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());

        dest2 = decForm.format(64000L, dest2, pos);
        System.out.println("dest2 = " + dest2);
        System.out.println("INTEGER fields are at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());
    }
}

/*
```

```
Output:
dest1 = 223423.00
INTEGER fields are at: 0, 6
dest2 = 64000.00
INTEGER fields are at: 0, 5
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.getDecimalFormatSymbols Method

Gets the DecimalFormatSymbols object that contains the symbols used when formatting a decimal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.DecimalFormatSymbols getDecimalFormatSymbols();
```

Return Value

The DecimalFormatSymbols object that contains the symbols used when formatting a decimal.

Example

The following example shows the various properties of the DecimalFormatSymbols object.

```
// decimalformat_getdecimalformatsymbols.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Display the various properties of the
        // DecimalFormatSymbols object.
        DecimalFormatSymbols dfs =
            decForm.getDecimalFormatSymbols();

        System.out.println("DecimalFormatSymbols.getDecimalSeparator: " +
            dfs.getDecimalSeparator());
        System.out.println("DecimalFormatSymbols.getDigit: " +
            dfs.getDigit());
        System.out.println("DecimalFormatSymbols.getGroupingSeparator: " +
            dfs.getGroupingSeparator());
        System.out.println("DecimalFormatSymbols.getInfinity: " +
            dfs.getInfinity());
        System.out.println("DecimalFormatSymbols.getMinusSign: " +
            dfs.getMinusSign());
        System.out.println("DecimalFormatSymbols.getNaN: " +
            dfs.getNaN());
        System.out.println("DecimalFormatSymbols.getPatternSeparator: " +
            dfs.getPatternSeparator());
        System.out.println("DecimalFormatSymbols.getPercent: " +
            dfs.getPercent());
        System.out.println("DecimalFormatSymbols.getPerMill: " +
            dfs.getPerMill());
        System.out.println("DecimalFormatSymbols.getZeroDigit: " +
            dfs.getZeroDigit());
    }
}

/*
Output:
DecimalFormatSymbols.getDecimalSeparator: .
DecimalFormatSymbols.getDigit: #
DecimalFormatSymbols.getGroupingSeparator: ,
DecimalFormatSymbols.getInfinity: 8
DecimalFormatSymbols.getMinusSign: -
DecimalFormatSymbols.getNaN: ?
```

```
DecimalFormatSymbols.getPatternSeparator: ;
DecimalFormatSymbols.getPercent: %
DecimalFormatSymbols.getPerMill: ë
DecimalFormatSymbols.getZeroDigit: 0
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.getGroupingSize Method

Gets the number of characters that are grouped together between grouping separators.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getGroupingSize();
```

Return Value

The number of characters that are grouped together between grouping separators.

Example

The following example displays the default grouping size for decimals.

```
// decimalformat_getgroupingsize.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Display the various properties of the DecimalFormat
        // object.
        System.out.println("Grouping Size: " +
            decForm.getGroupingSize());
    }
}

/*
Output:
Grouping Size: 3
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.hashCode Method

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

A hash code for the current object.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.getMultiplier Method

Gets the multiplier used for calculating percents or permills.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getMultiplier();
```

Return Value

The multiplier used for calculating percents or permills.

Example

The following example displays the default multiplier for decimals.

```
// decimalformat_getmultiplier.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Display the various properties of the DecimalFormat
        // object.
        System.out.println("Multiplier: " +
            decForm.getMultiplier());
    }
}

/*
Output:
Multiplier: 1
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.getNegativePrefix Method

Gets the symbol that is prefixed to a negative number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getNegativePrefix();
```

Return Value

The symbol that is prefixed to a negative number.

Example

The following example displays the default negative prefix for decimals.

```
// decimalformat_getnegativeprefix.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Display the various properties of the DecimalFormat
        // object.
        System.out.println("Negative Prefix: " +
            decForm.getNegativePrefix());
    }
}

/*
Output:
Negative Prefix: -
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.getNegativeSuffix Method

Gets the symbol that is appended to a negative number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getNegativeSuffix();
```

Return Value

The symbol that is appended to a negative number.

Example

The following example displays a common negative suffix for decimals.

```
// decimalformat_getnegativesuffix.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat("0.00;0.00-");

        // Display the various properties of the DecimalFormat
        // object.
        System.out.println("Negative Suffix: " +
            decForm.getNegativeSuffix());
    }
}

/*
Output:
Negative Suffix: -
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.getPositivePrefix Method

Gets the symbol that is prefixed to a positive number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getPositivePrefix();
```

Return Value

The symbol that is prefixed to a positive number.

Example

The following example displays a common positive prefix for decimals.

```
// decimalformat_getpositiveprefix.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat("+0.00");

        // Display the various properties of the DecimalFormat
        // object.
        System.out.println("Positive Prefix: " +
            decForm.getPositivePrefix());
    }
}

/*
Output:
Positive Prefix: +
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.getPositiveSuffix Method

Gets the symbol that is appended to a positive number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getPositiveSuffix();
```

Return Value

The symbol that is appended to a positive number.

Example

The following example displays a common positive suffix for decimals.

```
// decimalformat_getpositivesuffix.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat("0.00+");

        // Display the various properties of the DecimalFormat
        // object.
        System.out.println("Positive Suffix: " +
            decForm.getPositiveSuffix());
    }
}

/*
Output:
Positive Suffix: +
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.isDecimalSeparatorAlwaysShown Method

Determines whether the decimal separator is shown with integers.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean isDecimalSeparatorAlwaysShown();
```

Return Value

true if the decimal separator is shown with integers; false otherwise.

Example

The following example displays the default value that determines whether the decimal separator is shown for decimals.

```
// decimalformat_isdecialseparatoralwaysshown.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Display the various properties of the DecimalFormat
        // object.
        System.out.println("Is Decimal Separator Always Shown? " +
            decForm.isDecimalSeparatorAlwaysShown());
    }
}

/*
Output:
Is Decimal Separator Always Shown? false
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.parse Method

Parses a string into a decimal.

Overload List

Name	Description
DecimalFormat.parse (String)	Parses a string into a number represented by a Number object.
DecimalFormat.parse (String, ParsePosition)	Parses a string into a decimal represented by a Number object.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.parse Method (String, ParsePosition)

Parses a string into a decimal represented by a Number object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Number parse(
    java.lang.String str,
    java.text.ParsePosition pos);
```

Parameters

str

The string to be parsed into a decimal represented by a Number object.

pos

Used to indicate the first position in the string to be parsed.

Return Value

A Number object containing the string parsed into a decimal.

Example

The following example parses a string starting at position 8 into a number and displays the result to the Console.

```
// decimalformat_parse.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Parse the decimal starting at position 8.
        ParsePosition pos = new ParsePosition(8);
        Number n = decForm.parse("123456789.14", pos);

        System.out.println("Number = " + n);
        System.out.println("Position after parse = " +
            pos.getIndex());
    }
}

/*
Output:
Number = 9.14
Position after parse = 12
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.setDecimalFormatSymbols Method

Sets a DecimalFormatSymbols object that contains the symbols used when formatting a decimal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setDecimalFormatSymbols(
    java.text.DecimalFormatSymbols newSymbols);
```

Parameters

newSymbols

A DecimalFormatSymbols object that contains the symbols used when formatting a decimal.

Example

The following example sets the DecimalFormatSymbols object to a default instance of a DecimalFormatSymbols object.

```
// decimalformat_setdecimalformatsymbols.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Set the DecimalFormatSymbols to a default instance
        // of the DecimalFormatSymbols class.
        DecimalFormatSymbols dfs = new DecimalFormatSymbols();
        decForm.setDecimalFormatSymbols(dfs);

        // Display the various properties of the
        // DecimalFormatSymbols object.
        System.out.println("DecimalFormatSymbols.getDecimalSeparator: " +
            dfs.getDecimalSeparator());
        System.out.println("DecimalFormatSymbols.getDigit: " +
            dfs.getDigit());
        System.out.println("DecimalFormatSymbols.getGroupingSeparator: " +
            dfs.getGroupingSeparator());
        System.out.println("DecimalFormatSymbols.getInfinity: " +
            dfs.getInfinity());
        System.out.println("DecimalFormatSymbols.getMinusSign: " +
            dfs.getMinusSign());
        System.out.println("DecimalFormatSymbols.getNaN: " +
            dfs.getNaN());
        System.out.println("DecimalFormatSymbols.getPatternSeparator: " +
            dfs.getPatternSeparator());
        System.out.println("DecimalFormatSymbols.getPercent: " +
            dfs.getPercent());
        System.out.println("DecimalFormatSymbols.getPerMill: " +
            dfs.getPerMill());
        System.out.println("DecimalFormatSymbols.getZeroDigit: " +
            dfs.getZeroDigit());
    }
}

/*
Output:
DecimalFormatSymbols.getDecimalSeparator: .
DecimalFormatSymbols.getDigit: #
```

```
DecimalFormatSymbols.getGroupingSeparator: ,
DecimalFormatSymbols.getInfinity: 8
DecimalFormatSymbols.getMinusSign: -
DecimalFormatSymbols.getNaN: ?
DecimalFormatSymbols.getPatternSeparator: ;
DecimalFormatSymbols.getPercent: %
DecimalFormatSymbols.getPerMill: ë
DecimalFormatSymbols.getZeroDigit: 0
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.setDecimalSeparatorAlwaysShown Method

Sets a value indicating whether the decimal separator is shown with integers.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setDecimalSeparatorAlwaysShown(
    boolean newValue);
```

Parameters

newValue

true to display the decimal separator with integers; false otherwise.

Example

The following example sets a value indicating that the decimal separator should always be shown for decimals.

```
// decimalformat_setdecialseparatoralwaysshown.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Make the decimal separator always shown.
        decForm.setDecimalSeparatorAlwaysShown(true);
        System.out.println("Is Decimal Separator Always Shown? " +
            decForm.isDecimalSeparatorAlwaysShown());
    }
}

/*
Output:
Is Decimal Separator Always Shown? true
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.setGroupingSize Method

Sets the number of characters that are grouped together between grouping separators.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setGroupingSize(
    int groupSize);
```

Parameters

groupSize

The number of characters that are grouped together between grouping separators.

Example

The following example sets the grouping size for decimals to the value 3.

```
// decimalformat_setgroupingsize.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Set the grouping size to groups of 3.
        decForm.setGroupingSize(3);
        System.out.println("Grouping Size: " +
            decForm.getGroupingSize());
    }
}

/*
Output:
Grouping Size: 3
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.setMultiplier Method

Sets the multiplier used for calculating percents or permills.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setMultiplier(
    int newValue);
```

Parameters

newValue

The multiplier used for calculating percents or permills.

Example

The following example sets the multiplier for percents and permills to 2.

```
// decimalformat_setmultiplier.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Set the multiplier for percents and permills
        // to 2.
        decForm.setMultiplier(2);
        System.out.println("Multiplier: " +
            decForm.getMultiplier());
    }
}

/*
Output:
Multiplier: 2
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.setNegativePrefix Method

Sets the symbol that is prefixed to a negative number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setNegativePrefix(
    java.lang.String newValue);
```

Parameters

newValue

The symbol that is prefixed to a negative number.

Example

The following example sets the negative prefix and the negative suffix to "(" and ")", respectively.

```
// decimalformat_setnegativeprefix.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Set the negative prefix to "(" and the negative
        // suffix to ")".
        decForm.setNegativePrefix("(");
        decForm.setNegativeSuffix(")");
        System.out.println("Negative Prefix: " +
            decForm.getNegativePrefix());
        System.out.println("Negative Suffix: " +
            decForm.getNegativeSuffix());
    }
}

/*
Output:
Negative Prefix: (
Negative Suffix: )
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.setNegativeSuffix Method

Sets the symbol that is appended to a negative number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setNegativeSuffix(
    java.lang.String newValue);
```

Parameters

newValue

The symbol that is appended to a negative number.

Example

The following example sets the negative prefix and the negative suffix to "(" and ")", respectively.

```
// decimalformat_setnegativesuffix.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Set the negative prefix to "(" and the negative
        // suffix to ")".
        decForm.setNegativePrefix("(");
        decForm.setNegativeSuffix(")");
        System.out.println("Negative Prefix: " +
            decForm.getNegativePrefix());
        System.out.println("Negative Suffix: " +
            decForm.getNegativeSuffix());
    }
}

/*
Output:
Negative Prefix: (
Negative Suffix: )
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.setPositivePrefix Method

Sets the symbol that is prefixed to a positive number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setPositivePrefix(
    java.lang.String newValue);
```

Parameters

newValue

The symbol that is prefixed to a positive number.

Example

The following example sets the positive prefix to an empty string and the positive suffix to "+".

```
// decimalformat_setpositiveprefix.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Set the positive prefix to "" and the positive
        // suffix to "+".
        decForm.setPositivePrefix("");
        decForm.setPositiveSuffix("+");
        System.out.println("Positive Prefix: " +
            decForm.getPositivePrefix());
        System.out.println("Positive Suffix: " +
            decForm.getPositiveSuffix());
    }
}

/*
Output:
Positive Prefix:
Positive Suffix: +
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.setPositiveSuffix Method

Sets the symbol that is appended to a positive number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setPositiveSuffix(
    java.lang.String newValue);
```

Parameters

newValue

The symbol that is appended to a positive number.

Example

The following example sets the positive prefix to an empty string and the positive suffix to "+".

```
// decimalformat_setpositivesuffix.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();

        // Set the positive prefix to "" and the positive
        // suffix to "+".
        decForm.setPositivePrefix("");
        decForm.setPositiveSuffix("+");
        System.out.println("Positive Prefix: " +
            decForm.getPositivePrefix());
        System.out.println("Positive Suffix: " +
            decForm.getPositiveSuffix());
    }
}

/*
Output:
Positive Prefix:
Positive Suffix: +
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.toLocalizedPattern Method

Applies the previously set pattern to the string for an instance of a DecimalFormat object. Localization of the pattern is currently not supported by the J# Class Libraries.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toLocalizedPattern();
```

Return Value

The string representation of the decimal formatted using the previously set pattern.

Example

The following example displays the default localized pattern of a DecimalFormat object.

```
// decimalformat_tolocalizedpattern.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Display the default localized pattern for a
        // DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();
        System.out.println("Default Localized Pattern: " +
            decForm.toLocalizedPattern());
    }
}

/*
Output:
Default Localized Pattern: #,##0.###
*/
```

Remarks

Localization of the pattern is currently not supported by the J# Class Libraries. Calling this method has the same effect as calling toPattern.

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormat.toPattern Method

Applies the previously set pattern to the string for an instance of a DecimalFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toPattern();
```

Return Value

The string representation of the decimal formatted using the previously set pattern.

Example

The following example displays the default pattern of a DecimalFormat object.

```
// decimalformat_topattern.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Display the default localized pattern for a
        // DecimalFormat object.
        DecimalFormat decForm = new DecimalFormat();
        System.out.println("Default Pattern: " +
            decForm.toPattern());
    }
}

/*
Output:
Default Pattern: #,##0.###
*/
```

See Also

Reference

DecimalFormat Class

Concepts

DecimalFormat Members

java.text Package

DecimalFormatSymbols Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final class java.text.DecimalFormatSymbols
    extends java.lang.Object
    implements java.lang.Cloneable, java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

java.text.DecimalFormatSymbols

See Also

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols Members

The following tables list the members exposed by the DecimalFormatSymbols type.

Public Constructors

Name	Description
DecimalFormatSymbols	Overloaded.

Public Methods

Name	Description
equals	Overridden.
getDecimalSeparator	
getDigit	
getGroupingSeparator	
hashCode	Overridden.
getInfinity	
getMinusSign	
getNaN	
getPatternSeparator	
getPercent	
getPerMill	
getClass	(inherited from Object)
getZeroDigit	
MemberwiseClone	Performs a shallow copy of the members.
setDecimalSeparator	
setDigit	
setGroupingSeparator	
setInfinity	
setMinusSign	
setNaN	
setPatternSeparator	

setPercent	
setPerMill	
setZeroDigit	
toString	Overridden.

See Also

Reference

DecimalFormatSymbols Class

Concepts

java.text Package

DecimalFormatSymbols Constructor

Overload List

Name	Description
DecimalFormatSymbols ()	
DecimalFormatSymbols (Locale)	

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols Constructor ()

Initializes a new instance of the DecimalFormatSymbols Class .

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.DecimalFormatSymbols();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols Constructor (Locale)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.DecimalFormatSymbols(
    java.util.Locale loc);
```

Parameters

loc

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols Methods

Public Methods

Name	Description
equals	Overridden.
getDecimalSeparator	
getDigit	
getGroupingSeparator	
hashCode	Overridden.
getInfinity	
getMinusSign	
getNaN	
getPatternSeparator	
getPercent	
getPerMill	
getClass	(inherited from Object)
getZeroDigit	
MemberwiseClone	Performs a shallow copy of the members.
setDecimalSeparator	
setDigit	
setGroupingSeparator	
setInfinity	
setMinusSign	
setNaN	
setPatternSeparator	
setPercent	
setPerMill	
setZeroDigit	

toString	Overridden.

See Also

Reference

DecimalFormatSymbols Class

Concepts

java.text Package

DecimalFormatSymbols.clone Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.equals Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getDecimalSeparator Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char getDecimalSeparator();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getDigit Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char getDigit();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getGroupingSeparator Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char getGroupingSeparator();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.hashCode Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getInfinity Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getInfinity();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getMinusSign Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char getMinusSign();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getNaN Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getNaN();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getPatternSeparator Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char getPatternSeparator();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getPercent Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char getPercent();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getPerMill Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char getPerMill();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.getZeroDigit Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char getZeroDigit();
```

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.setDecimalSeparator Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setDecimalSeparator(
    char decimalSep);
```

Parameters
decimalSep

See Also
Reference
DecimalFormatSymbols Class
Concepts
DecimalFormatSymbols Members
java.text Package

DecimalFormatSymbols.setDigit Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setDigit(
    char digit);
```

Parameters
digit

See Also
Reference
DecimalFormatSymbols Class
Concepts
DecimalFormatSymbols Members
java.text Package

DecimalFormatSymbols.setGroupingSeparator Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setGroupingSeparator(
    char groupSep);
```

Parameters

groupSep

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.setInfinity Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setInfinity(
    java.lang.String infinity);
```

Parameters

infinity

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.setMinusSign Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setMinusSign(
    char minusSign);
```

Parameters

minusSign

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.setNaN Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setNaN(
    java.lang.String nan);
```

Parameters

nan

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.setPatternSeparator Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setPatternSeparator(
    char patternSep);
```

Parameters
patternSep

See Also
Reference
DecimalFormatSymbols Class
Concepts
DecimalFormatSymbols Members
java.text Package

DecimalFormatSymbols.setPercent Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setPercent(
    char percent);
```

Parameters

percent

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.setPerMill Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setPerMill(
    char perMill);
```

Parameters

perMill

See Also

Reference

DecimalFormatSymbols Class

Concepts

DecimalFormatSymbols Members

java.text Package

DecimalFormatSymbols.setZeroDigit Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setZeroDigit(
    char zeroDigit);
```

Parameters
zeroDigit

See Also
Reference
DecimalFormatSymbols Class
Concepts
DecimalFormatSymbols Members
java.text Package

FieldPosition Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public class java.text.FieldPosition
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.text.FieldPosition

See Also

Concepts

FieldPosition Members

java.text Package

FieldPosition Members

The following tables list the members exposed by the FieldPosition type.

Public Constructors

Name	Description
FieldPosition	

Public Methods

Name	Description
equals	(inherited from Object)
getBeginIndex	
getEndIndex	
getField	
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FieldPosition Class

Concepts

java.text Package

FieldPosition Constructor

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.FieldPosition(
    int field);
```

Parameters
field

See Also
Reference
FieldPosition Class
Concepts
FieldPosition Members
java.text Package

FieldPosition Methods

Public Methods

Name	Description
equals	(inherited from Object)
getBeginIndex	
getEndIndex	
getField	
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

FieldPosition Class

Concepts

java.text Package

FieldPosition.getBeginIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getBeginIndex();
```

See Also

Reference

FieldPosition Class

Concepts

FieldPosition Members

java.text Package

FieldPosition.getEndIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getEndIndex();
```

See Also

Reference

FieldPosition Class

Concepts

FieldPosition Members

java.text Package

FieldPosition.getField Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getField();
```

See Also

Reference

FieldPosition Class

Concepts

FieldPosition Members

java.text Package

Format Class

An abstract class used as the base class for the rest of the format classes in the java.text package.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.text.Format
    extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable
```

Example

The following example demonstrates the format and parseObject methods of the Format class. This example uses the NumberFormat class, which is a specialization of the abstract Format class.

```
// format_overview.jsl

import java.math.BigDecimal;
import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get a default NumberFormat instance.
            NumberFormat numForm = NumberFormat.getInstance();

            // Format some decimals using the pattern supplied above.
            BigDecimal bd1 = new BigDecimal(22.3423D);
            String dest1 = numForm.format(bd1);
            System.out.println("dest1 = " + dest1);

            BigDecimal bd2 = new BigDecimal(64000D);
            String dest2 = numForm.format(bd2);
            System.out.println("dest2 = " + dest2);

            // Parse the decimal string.
            Object o = numForm.parseObject("123456789.14");
            System.out.println("Number = " + o.toString());
        }
        catch (ParseException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
dest1 = 22.342
dest2 = 64,000
Number = 1.2345678914E8
*/
```

Inheritance Hierarchy

java.lang.Object

  java.text.Format

java.text.DateFormat

java.text.MessageFormat

java.text.NumberFormat

See Also

Concepts

Format Members

java.text Package

Format Members

An abstract class used as the base class for the rest of the format classes in the java.text package.

The following tables list the members exposed by the Format type.

Public Constructors

Name	Description
Format	Initializes a new instance of a Format object.

Public Methods

Name	Description
equals	(inherited from Object)
format	Overloaded. Formats an object into a string.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a new instance of a Format object that is a shallow copy of this instance.
parseObject	Overloaded. Parses a string into an Object.
toString	Overridden. Displays a human-readable representation of a Format object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Format Class

Concepts

java.text Package

Format Constructor

Initializes a new instance of a Format object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.Format();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Format Class

Concepts

Format Members

java.text Package

Format Methods

Public Methods

Name	Description
equals	(inherited from Object)
format	Overloaded. Formats an object into a string.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	Creates a new instance of a Format object that is a shallow copy of this instance.
parseObject	Overloaded. Parses a string into an Object.
toString	Overridden. Displays a human-readable representation of a Format object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Format Class

Concepts

java.text Package

Format.clone Method

Creates an instance of a Format object that is a deep copy of the existing object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

Return Value

An instance of a Format object that is a deep copy of the existing object.

See Also

Reference

Format Class

Concepts

Format Members

java.text Package

Format.format Method

Formats an object into a string.

Overload List

Name	Description
Format.format (Object)	Formats an object into a string.
Format.format (Object, StringBuffer, FieldPosition)	Formats an object and appends it to a StringBuffer object.

See Also

Reference

Format Class

Concepts

Format Members

java.text Package

Format.format Method (Object)

Formats an object into a string.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String format(
    java.lang.Object obj);
```

Parameters

obj

The object to be formatted into a string.

Return Value

The object formatted into a string.

Example

The following example formats a BigDecimal object. This example uses the NumberFormat class, which is a specialization of the abstract Format class.

```
// format_format.jsl

import java.math.BigDecimal;
import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some decimals using the pattern supplied above.
        BigDecimal bd1 = new BigDecimal(22.3423D);
        String dest1 = numForm.format(bd1);
        System.out.println("dest1 = " + dest1);

        BigDecimal bd2 = new BigDecimal(64000D);
        String dest2 = numForm.format(bd2);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 22.342
dest2 = 64,000
*/
```

See Also

Reference

Format Class

Concepts

Format Members

java.text Package

Format.format Method (Object, StringBuffer, FieldPosition)

Formats an object and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.StringBuffer format(
    java.lang.Object obj,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition pos);
```

Parameters

obj

The object to be formatted into a string.

appendBuf

A StringBuffer object to be appended with the given object formatted into a string.

pos

Used to find start and end indices of a particular field of the date in the formatted string.

Return Value

A StringBuffer object appended with the formatted object provided.

Example

The following example formats a BigDecimal object and prints out the beginning and ending indices of the integer part of the BigDecimal object. This example uses the NumberFormat class, which is a specialization of the abstract Format class.

```
// format_format_2.jsl

import java.math.BigDecimal;
import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.INTEGER_FIELD);

        BigDecimal bd1 = new BigDecimal(22.3423D);
        dest1 = numForm.format(bd1, dest1, pos);
        System.out.println("dest1 = " + dest1);
        System.out.println("INTEGER portion is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());

        BigDecimal bd2 = new BigDecimal(64000D);
        dest2 = numForm.format(bd2, dest2, pos);
        System.out.println("dest2 = " + dest2);
        System.out.println("INTEGER portion is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());
    }
}
```

```
/*
Output:
dest1 = 22.342
INTEGER portion is at: 0, 2
dest2 = 64,000
INTEGER portion is at: 0, 6
*/
```

See Also

Reference

Format Class

Concepts

Format Members

java.text Package

Format.parseObject Method

Parses a string into an Object.

Overload List

Name	Description
Format.parseObject (String)	Parses a string into an Object.
Format.parseObject (String, ParsePosition)	Parses a string into an Object.

See Also

Reference

Format Class

Concepts

Format Members

java.text Package

Format.parseObject Method (String)

Parses a string into an Object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object parseObject(
    java.lang.String sourceStr) throws java.text.ParseException;
```

Parameters

sourceStr

The string to be parsed into an Object.

Return Value

An Object containing the date represented by the string.

Example

The following example parses a string into an Object and displays the result to the Console. This example uses the NumberFormat class, which is a specialization of the abstract Format class.

```
// format_parseobject.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get a default NumberFormat instance.
            NumberFormat numForm = NumberFormat.getInstance();

            // Parse the decimal string.
            Object o = numForm.parseObject("123456789.14");
            System.out.println("Number = " + o.toString());
        }
        catch (ParseException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Number = 1.2345678914E8
*/
```

See Also

Reference

Format Class

Concepts

Format Members

java.text Package

Format.parseObject Method (String, ParsePosition)

Parses a string into an Object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object parseObject(
    java.lang.String sourceStr,
    java.text.ParsePosition pos);
```

Parameters

sourceStr

The string to be parsed into an Object.

pos

Used to indicate the first position in the string to be parsed.

Return Value

An Object containing the date represented by the string.

Example

The following example parses a string starting at position 8 into an Object and displays the result to the Console. This example uses the NumberFormat class, which is a specialization of the abstract Format class.

```
// format_parseobject_2.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Parse the decimal starting at position 8.
        ParsePosition pos = new ParsePosition(8);
        Object o = numForm.parseObject("123456789.14", pos);
        System.out.println("Number = " + o.toString());
    }
}

/*
Output:
Number = 9.14
*/
```

See Also

Reference

Format Class

Concepts

Format Members

java.text Package

MessageFormat Class

Contains methods and properties used to format a message as a string and to parse a string into a message. A message is simply a collection of objects that are formatted according to their type.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public class java.text.MessageFormat
    extends java.text.Format
```

Example

The following example shows how to perform common operations using a MessageFormat object, such as formatting, parsing, and applying a pattern and locale.

```
// messageformat_overview.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.applyPattern(pattern);
        Object[] values = null;
        try
        {
            values = mFmt.parse(
                "I bought 20 apples for $4.18 on 11-Nov-2004.");
        }
        catch (ParseException ex)
        {
            System.out.println(ex.toString());
        }

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(values));
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,dd-MMM-yyyy}.
I bought 20 apples for $4.18 on 11-Nov-2004.
*/
```

Inheritance Hierarchy

java.lang.Object

  java.text.Format

java.text.MessageFormat

See Also

Concepts

MessageFormat Members

java.text Package

MessageFormat Members

Contains methods and properties used to format a message as a string and to parse a string into a message. A message is simply a collection of objects that are formatted according to their type.

The following tables list the members exposed by the MessageFormat type.

Public Constructors

Name	Description
MessageFormat	Initializes a new instance of a MessageFormat object. The message will be formatted into a string using the pattern provided.

Public Methods

Name	Description
applyPattern	Changes the pattern that an instance of a MessageFormat object is using after it is created.
equals	Overridden. Determines whether two MessageFormat objects are equal.
format	Overloaded. Formats a message into a string.
getFormats	Gets the array of Format objects that are used on variables in the pattern.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getLocale	Gets the Locale object that is being used to format and parse messages.
getClass	(inherited from Object)
clone	Creates a shallow copy of the current MessageFormat object.
parse	Overloaded. Parses a string into an array of objects.
parseObject	Overloaded. Parses a string into an object.
setFormat	Sets the Format object to be used on a variable in the pattern.
setFormats	Sets all the Format objects to be used on variables in a pattern.
setLocale	Sets the Locale object that is being used to format and parse messages.
toPattern	Applies the previously set pattern to the string for an instance of a MessageFormat object.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

MessageFormat Class

Concepts

java.text Package

MessageFormat Constructor

Initializes a new instance of a MessageFormat object. The message will be formatted into a string using the pattern provided.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.MessageFormat(
    java.lang.String pattern);
```

Parameters

pattern

The pattern describing how to format the message as a string.

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat Methods

Public Methods

Name	Description
applyPattern	Changes the pattern that an instance of a MessageFormat object is using after it is created.
equals	Overridden. Determines whether two MessageFormat objects are equal.
format	Overloaded. Formats a message into a string.
getFormats	Gets the array of Format objects that are used on variables in the pattern.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getLocale	Gets the Locale object that is being used to format and parse messages.
getClass	(inherited from Object)
clone	Creates a shallow copy of the current MessageFormat object.
parse	Overloaded. Parses a string into an array of objects.
parseObject	Overloaded. Parses a string into an object.
setFormat	Sets the Format object to be used on a variable in the pattern.
setFormats	Sets all the Format objects to be used on variables in a pattern.
setLocale	Sets the Locale object that is being used to format and parse messages.
toPattern	Applies the previously set pattern to the string for an instance of a MessageFormat object.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

MessageFormat Class

Concepts

java.text Package

MessageFormat.applyPattern Method

Changes the pattern that an instance of a MessageFormat object is using after it is created.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void applyPattern(
    java.lang.String newPattern);
```

Parameters

newPattern

Specifies the style used to format to or parse from message strings.

Example

The following example shows how to apply a pattern to a MessageFormat object.

```
// messageformat_applypattern.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Specify which formatters are to be used for each
        // of the placeholders in the pattern above.
        Format[] formats = { new DecimalFormat("#"),
            NumberFormat.getCurrencyInstance(),
            new SimpleDateFormat("MMM/dd/yyyy") };

        // Create values to populate the position in the pattern.
        Object[] values = { new Integer(5), new Double(7.53),
            new Date("04-Jul-2004") };

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.setFormats(formats);
        mFmt.applyPattern(pattern);

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(values));
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,dd-MMM-yyyy}.
I bought 5 apples for $7.53 on 04-Jul-2004.
*/
```

See Also

Reference
MessageFormat Class
Concepts
MessageFormat Members
java.text Package

MessageFormat.clone Method

Creates a shallow copy of the current MessageFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object clone();
```

Return Value

A shallow copy of the current MessageFormat object.

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.equals Method

Determines whether two MessageFormat objects are equal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

A MessageFormat object to compare with the current MessageFormat object for equality.

Return Value

true if the two MessageFormat objects are equal; false otherwise.

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.format Method

Formats a message into a string.

Overload List

Name	Description
MessageFormat.format (Object)	Formats an object into a string.
MessageFormat.format (String, Object[])	Formats a message, represented as an array of objects, into a string using the given pattern.
MessageFormat.format (Object, StringBuffer, FieldPosition)	Formats a message, represented as a single object, and appends it to a StringBuffer object.
MessageFormat.format (Object[], StringBuffer, FieldPosition)	Formats a message, represented as an array of objects, and appends it to a StringBuffer object.

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.format Method (String, Object[ ])

Formats a message, represented as an array of objects, into a string using the given pattern.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.String format(
    java.lang.String pattern,
    java.lang.Object[] arguments);
```

Parameters

pattern

Specifies the style used to format to or parse from message strings.

arguments

The objects to be formatted into a string.

Return Value

A string containing the formatted message.

Example

The following example shows how to display a formatted string using the MessageFormat object.

```
// messageformat_format1.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Specify which formatters are to be used for each
        // of the placeholders in the pattern above.
        Format[] formats = { new DecimalFormat("#"),
            NumberFormat.getCurrencyInstance(),
            new SimpleDateFormat("MMM/dd/yyyy") };

        // Create values to populate the position in the pattern.
        Object[] values = { new Integer(5), new Double(7.53),
            new Date("04-Jul-2004") };

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.setFormats(formats);

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(pattern, values));
    }
}

/*
```

```
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,MMM/dd/yyyy}.
I bought 5 apples for $7.53 on 04-Jul-2004.
*/
```

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.format Method (Object, StringBuffer, FieldPosition)

Formats a message, represented as a single object, and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.StringBuffer format(
    java.lang.Object singleArg,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition ignore);
```

Parameters

singleArg

The object to be formatted into a string.

appendBuf

A StringBuffer object to be appended with the object formatted into a string.

ignore

Unused.

Return Value

A StringBuffer object appended with the object formatted into a string.

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.format Method (Object[ ], StringBuffer, FieldPosition)

Formats a message, represented as an array of objects, and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.StringBuffer format(
    java.lang.Object[] arguments,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition ignore);
```

Parameters

arguments

The objects to be formatted into a string.

appendBuf

A StringBuffer object to be appended with the objects formatted into a string.

ignore

Unused.

Return Value

A StringBuffer object appended with the objects formatted into a string.

Example

The following example shows how to display a formatted string using the MessageFormat object.

```
// messageformat_format3.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Specify which formatters are to be used for each
        // of the placeholders in the pattern above.
        Format[] formats = { new DecimalFormat("#"),
            NumberFormat.getCurrencyInstance(),
            new SimpleDateFormat("MMM/dd/yyyy") };

        // Create values to populate the position in the pattern.
        Object[] values = { new Integer(5), new Double(7.53),
            new Date("04-Jul-2004") };

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.setFormats(formats);

        // Print out the pattern being used for formatting
```

```
            // and the formatted output.
            System.out.println(mFmt.toPattern());
            StringBuffer buf = new StringBuffer();
            System.out.println(mFmt.format(values, buf, null));
        }
    }

    /*
    Output:
    I bought {0,number,#} apples for {1,number,currency} on {2,date,MMM/dd/yyyy}.
    I bought 5 apples for $7.53 on Jul/04/2004.
    */
```

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.getFormats Method

Gets the array of Format objects that are used on variables in the pattern.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.Format[] getFormats();
```

Return Value

The array of Format objects that are used on variables in the pattern.

Example

The following example shows how to get the formatters that were used on a MessageFormat object.

```
// messageformat_getformats.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Specify which formatters are to be used for each
        // of the placeholders in the pattern above.
        Format[] formats = { new DecimalFormat("#"),
            NumberFormat.getCurrencyInstance(),
            new SimpleDateFormat("MMM/dd/yyyy") };

        // Create values to populate the position in the pattern.
        Object[] values = { new Integer(5), new Double(7.53),
            new Date("04-Jul-2004") };

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.setFormats(formats);

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(pattern, values));
        Format[] newFormats = mFmt.getFormats();
        for (int i = 0; i < newFormats.length; i++)
        {
            System.out.println(newFormats[i].toString());
        }
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,MMM/dd/yyyy}.
I bought 5 apples for $7.53 on 04-Jul-2004.
java.text.DecimalFormat@28d6c
java.text.DecimalFormat@29380
```

```
java.text.SimpleDateFormat@dca39d3c
*/
```

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.hashCode Method

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

A hash code for the current object.

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.getLocale Method

Gets the Locale object that is being used to format and parse messages.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.util.Locale getLocale();
```

Return Value

The Locale object that is being used to format and parse messages.

Example

The following example shows how to get the Locale that is being used to format and parse messages.

```
// messageformat_getlocale.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Specify which formatters are to be used for each
        // of the placeholders in the pattern above.
        Format[] formats = { new DecimalFormat("#"),
            NumberFormat.getCurrencyInstance(),
            new SimpleDateFormat("MMM/dd/yyyy") };

        // Create values to populate the position in the pattern.
        Object[] values = { new Integer(5), new Double(7.53),
            new Date("04-Jul-2004") };

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.setFormats(formats);
        mFmt.applyPattern(pattern);

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(values));
        System.out.println("Locale = " + mFmt.getLocale());
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,dd-MMM-yyyy}.
I bought 5 apples for $7.53 on 04-Jul-2004.
Locale = en_US
*/
```

See Also

Reference
MessageFormat Class
Concepts
MessageFormat Members
java.text Package

MessageFormat.parse Method

Parses a string into an array of objects.

Overload List

Name	Description
MessageFormat.parse (String)	Parses a string into an array of objects.
MessageFormat.parse (String, ParsePosition)	Parses a string into an array of objects.

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.parse Method (String)

Parses a string into an array of objects.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object[] parse(
    java.lang.String sourceStr) throws java.text.ParseException;
```

Parameters

sourceStr

The string to be parsed into an array of objects.

Return Value

The objects representing the various parts of the message string.

Example

The following example shows how to parse a string into an array of objects.

```
// messageformat_parse1.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        Object[] values = null;
        try
        {
            values = mFmt.parse(
                "I bought 20 apples for $4.18 on 11-Nov-2004.");
        }
        catch (ParseException ex)
        {
            System.out.println(ex.toString());
        }

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(values));
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,dd-MMM-yyyy}.
I bought 20 apples for $4.18 on 11-Nov-2004.
*/
```

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.parse Method (String, ParsePosition)

Parses a string into an array of objects.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object[] parse(
    java.lang.String sourceStr,
    java.text.ParsePosition pos);
```

Parameters

sourceStr

The string to be parsed into an array of objects.

pos

Used to indicate the first position in the string to be parsed.

Return Value

The objects representing the various parts of the message string.

Example

The following example shows how to parse a string into an array of objects.

```
// messageformat_parse2.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        Object[] values = mFmt.parse(
            "I bought 20 apples for $4.18 on 11-Nov-2004.",
            new ParsePosition(0));

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(values));
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,dd-MMM-yyyy}.
I bought 20 apples for $4.18 on 11-Nov-2004.
*/
```

See Also

Reference

MessageFormat Class
Concepts
MessageFormat Members
java.text Package

MessageFormat.parseObject Method

Parses a string into an object.

Overload List

Name	Description
MessageFormat.parseObject (String)	Parses a string into an Object.
MessageFormat.parseObject (String, ParsePosition)	Parses a string into an array of objects.

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.parseObject Method (String, ParsePosition)

Parses a string into an array of objects.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object parseObject(
    java.lang.String sourceStr,
    java.text.ParsePosition pos);
```

Parameters

sourceStr

The string to be parsed into an object.

pos

Used to indicate the first position in the string to be parsed.

Return Value

An object representing the parsed message string.

Example

The following example shows how to parse a string into an array of objects.

```
// messageformat_parseobject.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        Object value = mFmt.parseObject(
            "I bought 20 apples for $4.18 on 11-Nov-2004.",
            new ParsePosition(0));

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(value));
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,dd-MMM-yyyy}.
I bought 20 apples for $4.18 on 11-Nov-2004.
*/
```

See Also

Reference

MessageFormat Class
Concepts
MessageFormat Members
java.text Package

MessageFormat.setFormat Method

Sets the Format object to be used on a variable in the pattern.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setFormat(
    int variableNum,
    java.text.Format newFormat);
```

Parameters

variableNum

The index within the pattern that this Format object applies to.

newFormat

The Format object to be used on a variable in the pattern.

Example

The following example shows how to set the Format object to use for a position in the MessageFormat object.

```
// messageformat_setformat.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Specify which formatters are to be used for each
        // of the placeholders in the pattern above.
        Format[] formats = { new DecimalFormat("#"),
            NumberFormat.getCurrencyInstance(),
            new SimpleDateFormat("MMM/dd/yyyy") };

        // Create values to populate the position in the pattern.
        Object[] values = { new Integer(5), new Double(7.53),
            new Date("04-Jul-2004") };

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.setFormat(0, formats[0]);
        mFmt.setFormat(1, formats[1]);
        mFmt.setFormat(2, formats[2]);

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(pattern, values));
    }
}

/*
Output:
```

```
  I bought {0,number,#} apples for {1,number,currency} on {2,date,MMM/dd/yyyy}.
  I bought 5 apples for $7.53 on 04-Jul-2004.
  */
```

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.setFormats Method

Sets all the Format objects to be used on variables in a pattern.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setFormats(
    java.text.Format[] newFormats);
```

Parameters

newFormats

An array of Format objects to be used on variables in the pattern.

Example

The following example shows how to set the Format objects to use for all positions in the MessageFormat object.

```
// messageformat_setformats.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Specify which formatters are to be used for each
        // of the placeholders in the pattern above.
        Format[] formats = { new DecimalFormat("#"),
            NumberFormat.getCurrencyInstance(),
            new SimpleDateFormat("MMM/dd/yyyy") };

        // Create values to populate the position in the pattern.
        Object[] values = { new Integer(5), new Double(7.53),
            new Date("04-Jul-2004") };

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.setFormats(formats);

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(pattern, values));
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,MMM/dd/yyyy}.
I bought 5 apples for $7.53 on 04-Jul-2004.
*/
```

See Also

Reference

MessageFormat Class
Concepts
MessageFormat Members
java.text Package

MessageFormat.setLocale Method

Sets the Locale object that is being used to format and parse messages.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setLocale(
    java.util.Locale loc);
```

Parameters

loc

The Locale object that is being used to format and parse messages.

Example

The following example shows how to change the default locale used for formatting and parsing MessageFormat objects.

```
// messageformat_setlocale.jsl

import java.text.*;
import java.util.Date;
import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Specify which formatters are to be used for each
        // of the placeholders in the pattern above.
        Format[] formats = { new DecimalFormat("#"),
            NumberFormat.getCurrencyInstance(),
            new SimpleDateFormat("MMM/dd/yyyy") };

        // Create values to populate the position in the pattern.
        Object[] values = { new Integer(5), new Double(7.53),
            new Date("04-Jul-2004") };

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.setFormats(formats);
        mFmt.applyPattern(pattern);
        mFmt.setLocale(Locale.GERMANY);

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(values));
        System.out.println("Locale = " + mFmt.getLocale());
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,dd-MMM-yyyy}.
I bought 5 apples for 7,53â,¬ on 04-Jul-200
Locale = de_DE
```

```
    */
```

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members

java.text Package

MessageFormat.toPattern Method

Applies the previously set pattern to the string for an instance of a MessageFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toPattern();
```

Return Value

The string representation of the objects comprising the message formatted using the previously set pattern.

Example

The following example shows how to display the pattern being used to format MessageFormat objects.

```
// messageformat_topattern.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a pattern for our MessageFormat object to use.
        String pattern = "I bought {0,number,#} " +
            "apples for {1,number,currency} " +
            "on {2,date,dd-MMM-yyyy}.";

        // Specify which formatters are to be used for each
        // of the placeholders in the pattern above.
        Format[] formats = { new DecimalFormat("#"),
            NumberFormat.getCurrencyInstance(),
            new SimpleDateFormat("MMM/dd/yyyy") };

        // Create values to populate the position in the pattern.
        Object[] values = { new Integer(5), new Double(7.53),
            new Date("04-Jul-2004") };

        // Create a MessageFormat object and apply the pattern
        // to it.
        MessageFormat mFmt = new MessageFormat(pattern);
        mFmt.setFormats(formats);

        // Print out the pattern being used for formatting
        // and the formatted output.
        System.out.println(mFmt.toPattern());
        System.out.println(mFmt.format(pattern, values));
    }
}

/*
Output:
I bought {0,number,#} apples for {1,number,currency} on {2,date,MMM/dd/yyyy}.
I bought 5 apples for $7.53 on 04-Jul-2004.
*/
```

See Also

Reference

MessageFormat Class

Concepts

MessageFormat Members
java.text Package

NumberFormat Class

Contains methods and properties used to format a number as a string and to parse a string into a number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.text.NumberFormat
    extends java.text.Format
    implements java.lang.Cloneable
```

Example

The following example demonstrates the format, FRACTION_FIELD, getInstance, INTEGER_FIELD, and parse methods of the NumberFormat class.

```
// numberformat_overview.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get a default NumberFormat instance.
            NumberFormat numForm = NumberFormat.getInstance();

            // Format some decimals using the pattern supplied above.
            StringBuffer dest1 = new StringBuffer(24);
            StringBuffer dest2 = new StringBuffer(24);
            FieldPosition pos1 = new FieldPosition(NumberFormat.INTEGER_FIELD);
            FieldPosition pos2 = new FieldPosition(NumberFormat.FRACTION_FIELD);

            dest1 = numForm.format(22.3423D, dest1, pos1);
            System.out.println("dest1 = " + dest1);
            System.out.println("INTEGER is at: " + pos1.getBeginIndex() +
                ", " + pos1.getEndIndex());

            dest2 = numForm.format(64000D, dest2, pos2);
            System.out.println("dest2 = " + dest2);
            System.out.println("FRACTION is at: " + pos2.getBeginIndex() +
                ", " + pos2.getEndIndex());

            // Parse a decimal value.
            Number n = numForm.parse("123456789.14");
            System.out.println("Number = " + n);
        }
        catch (ParseException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
dest1 = 22.342
INTEGER is at: 0, 2
dest2 = 64,000
FRACTION is at: 6, 6
Number = 1.2345678914E8
```

```
    */
```

Inheritance Hierarchy

java.lang.Object

  java.text.Format

   java.text.NumberFormat

     java.text.ChoiceFormat

     java.text.DecimalFormat

See Also

Concepts

NumberFormat Members

java.text Package

NumberFormat Members

Contains methods and properties used to format a number as a string and to parse a string into a number.

The following tables list the members exposed by the NumberFormat type.

Public Fields

Name	Description
FRACTION_FIELD	A value used to refer to the fractional portion of a number.
INTEGER_FIELD	A value used to refer to the integer portion of a number.

Public Methods

Name	Description
equals	Overridden. Determines whether two NumberFormat objects are equal.
format	Overloaded. Formats a number into a string.
getAvailableLocales	Gets an array of all Locale objects that are supported by the NumberFormat class.
getCurrencyInstance	Overloaded. Creates an instance of a NumberFormat object that can be used to format and parse currency values.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getInstance	Overloaded. Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.
getMaximumFractionDigits	Gets a value representing the maximum number of digits displayed in the fractional part of the number.
getMaximumIntegerDigits	Gets a value representing the maximum number of digits displayed in the integer part of the number.
getMinimumFractionDigits	Gets a value representing the minimum number of digits displayed in the fractional part of the number.
getMinimumIntegerDigits	Gets a value representing the minimum number of digits displayed in the integer part of the number.
getNumberInstance	Overloaded. Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.
getPercentInstance	Overloaded. Creates an instance of a NumberFormat object and sets the pattern to the one used to format and parse percentage values.
getClass	(inherited from Object)
isGroupingUsed	Determines whether digits are being grouped together.
isParseIntegerOnly	Determines whether only the integer portion of the number is being parsed or if the fractional part is also included.

clone	Creates a shallow copy of the current NumberFormat object.
parse	Overloaded. Parses a string into a number.
parseObject	Overloaded. Parses a string into an Object.
setGroupingUsed	Sets a value determining whether digits are being grouped together.
setMaximumFractionDigits	Sets the value representing the maximum number of digits displayed in the fractional part of the number.
setMaximumIntegerDigits	Sets the value representing the maximum number of digits displayed in the integer part of the number.
setMinimumFractionDigits	Sets the value representing the minimum number of digits displayed in the fractional part of the number.
setMinimumIntegerDigits	Sets the value representing the minimum number of digits displayed in the integer part of the number.
setParseIntegerOnly	Sets a value determining whether only the integer portion of the number is being parsed or if the fractional part is also included.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NumberFormat Class

Concepts

java.text Package

NumberFormat Fields

Public Fields

Name	Description
FRACTION_FIELD	A value used to refer to the fractional portion of a number.
INTEGER_FIELD	A value used to refer to the integer portion of a number.

See Also

Reference

NumberFormat Class

Concepts

java.text Package

NumberFormat.FRACTION_FIELD Field

A value used to refer to the fractional portion of a number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int FRACTION_FIELD;
```

Example

The following example shows how to format a number and display the indices of the fractional part of the number.

```
// numberformat_fraction_field.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.FRACTION_FIELD);

        dest1 = numForm.format(22.3423D, dest1, pos);
        System.out.println("dest1 = " + dest1);
        System.out.println("FRACTION is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());

        dest2 = numForm.format(64000D, dest2, pos);
        System.out.println("dest2 = " + dest2);
        System.out.println("FRACTION is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());
    }
}

/*
Output:
dest1 = 22.342
FRACTION is at: 3, 6
dest2 = 64,000
FRACTION is at: 6, 6
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.INTEGER_FIELD Field

A value used to refer to the integer portion of a number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final int INTEGER_FIELD;
```

Example

The following example shows how to format a number and display the indices of the integer part of the number.

```
// numberformat_integer_field.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.INTEGER_FIELD);

        dest1 = numForm.format(22.3423D, dest1, pos);
        System.out.println("dest1 = " + dest1);
        System.out.println("INTEGER is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());

        dest2 = numForm.format(64000D, dest2, pos);
        System.out.println("dest2 = " + dest2);
        System.out.println("INTEGER is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());
    }
}

/*
Output:
dest1 = 22.342
INTEGER is at: 0, 2
dest2 = 64,000
INTEGER is at: 0, 6
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat Methods

Public Methods

Name	Description
equals	Overridden. Determines whether two NumberFormat objects are equal.
format	Overloaded. Formats a number into a string.
getAvailableLocales	Gets an array of all Locale objects that are supported by the NumberFormat class.
getCurrencyInstance	Overloaded. Creates an instance of a NumberFormat object that can be used to format and parse currency values.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getInstance	Overloaded. Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.
getMaximumFractionDigits	Gets a value representing the maximum number of digits displayed in the fractional part of the number.
getMaximumIntegerDigits	Gets a value representing the maximum number of digits displayed in the integer part of the number.
getMinimumFractionDigits	Gets a value representing the minimum number of digits displayed in the fractional part of the number.
getMinimumIntegerDigits	Gets a value representing the minimum number of digits displayed in the integer part of the number.
getNumberInstance	Overloaded. Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.
getPercentInstance	Overloaded. Creates an instance of a NumberFormat object and sets the pattern to the one used to format and parse percentage values.
getClass	(inherited from Object)
isGroupingUsed	Determines whether digits are being grouped together.
isParseIntegerOnly	Determines whether only the integer portion of the number is being parsed or if the fractional part is also included.
clone	Creates a shallow copy of the current NumberFormat object.
parse	Overloaded. Parses a string into a number.
parseObject	Overloaded. Parses a string into an Object.
setGroupingUsed	Sets a value determining whether digits are being grouped together.

setMaximumFractionDigits	Sets the value representing the maximum number of digits displayed in the fractional part of the number.
setMaximumIntegerDigits	Sets the value representing the maximum number of digits displayed in the integer part of the number.
setMinimumFractionDigits	Sets the value representing the minimum number of digits displayed in the fractional part of the number.
setMinimumIntegerDigits	Sets the value representing the minimum number of digits displayed in the integer part of the number.
setParseIntegerOnly	Sets a value determining whether only the integer portion of the number is being parsed or if the fractional part is also included.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NumberFormat Class

Concepts

java.text Package

NumberFormat.clone Method

Creates a shallow copy of the current NumberFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object clone();
```

Return Value

A shallow copy of the current NumberFormat object.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.equals Method

Determines whether two NumberFormat objects are equal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

A NumberFormat object to compare with the current NumberFormat object for equality.

Return Value

true if the two NumberFormat objects are equal; false otherwise.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.format Method

Formats a number into a string.

Overload List

Name	Description
NumberFormat.format (double)	Formats a double into a string.
NumberFormat.format (long)	Formats a long into a string.
NumberFormat.format (Object)	Formats an object into a string.
NumberFormat.format (double, StringBuffer, FieldPosition)	Formats a double and appends it to a StringBuffer object.
NumberFormat.format (long, StringBuffer, FieldPosition)	Formats a long and appends it to a StringBuffer object.
NumberFormat.format (Object, StringBuffer, FieldPosition)	Formats an Object and appends it to a StringBuffer object.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.format Method (Double)

Formats a double into a string.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String format(
    double number);
```

Parameters

number

The double to be formatted into a string.

Return Value

A string containing the value of the double.

Example

The following example formats a decimal value.

```
// numberformat_format.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some decimals using the pattern supplied above.
        String dest1 = numForm.format(22.3423D);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000D);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 22.342
dest2 = 64,000
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.format Method (Int64)

Formats a long into a string.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String format(
    long number);
```

Parameters

number

The long to be formatted into a string.

Return Value

A string containing the value of the long.

Example

The following example formats a long as a decimal.

```
// numberformat_format_2.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some longs using the pattern supplied above.
        String dest1 = numForm.format(223423L);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000L);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 223,423
dest2 = 64,000
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.format Method (Double, StringBuffer, FieldPosition)

Formats a double and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.StringBuffer format(
    double number,
    java.lang.StringBuffer appendBuffer,
    java.text.FieldPosition pos);
```

Parameters

number

The double to be formatted into a string.

appendBuffer

A StringBuffer object to be appended with the double formatted into a string.

pos

Used to find start and end indices of a particular field of the number in the formatted string.

Return Value

A StringBuffer object appended with the double formatted into a string.

Example

The following example formats a decimal value and prints out the beginning and ending indices of the fraction part of the decimal.

```
// numberformat_format_3.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.FRACTION_FIELD);

        dest1 = numForm.format(22.3423D, dest1, pos);
        System.out.println("dest1 = " + dest1);
        System.out.println("FRACTION is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());

        dest2 = numForm.format(64000D, dest2, pos);
        System.out.println("dest2 = " + dest2);
        System.out.println("FRACTION is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());
    }
}

/*
```

```
Output:
dest1 = 22.342
FRACTION is at: 3, 6
dest2 = 64,000
FRACTION is at: 6, 6
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.format Method (Int64, StringBuffer, FieldPosition)

Formats a long and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.StringBuffer format(
    long number,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition pos);
```

Parameters

number

The long to be formatted into a string.

appendBuf

A StringBuffer object to be appended with the long formatted into a string.

pos

Used to find start and end indices of a particular field of the number in the formatted string.

Return Value

A StringBuffer object appended with the long formatted into a string.

Example

The following example formats a long value and prints out the beginning and ending indices of the integer part of the long.

```
// numberformat_format_4.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some longs using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.INTEGER_FIELD);

        dest1 = numForm.format(223423L, dest1, pos);
        System.out.println("dest1 = " + dest1);
        System.out.println("INTEGER portion is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());

        dest2 = numForm.format(64000L, dest2, pos);
        System.out.println("dest2 = " + dest2);
        System.out.println("INTEGER portion is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());
    }
}

/*
Output:
```

```
dest1 = 223,423
INTEGER portion is at: 0, 7
dest2 = 64,000
INTEGER portion is at: 0, 6
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.format Method (Object, StringBuffer, FieldPosition)

Formats an Object and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.StringBuffer format(
    java.lang.Object number,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition pos);
```

Parameters

number

The Object to be formatted into a string.

appendBuf

A StringBuffer object to be appended with the Object formatted into a string.

pos

Used to find start and end indices of a particular field of the number in the formatted string.

Return Value

A StringBuffer object appended with the Object formatted into a string.

Example

The following example formats a **java.lang.BigDecimal** object and prints out the beginning and ending indices of the integer part of the BigDecimal object.

```
// numberformat_format_5.jsl

import java.math.BigDecimal;
import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some decimals using the pattern supplied above.
        StringBuffer dest1 = new StringBuffer(24);
        StringBuffer dest2 = new StringBuffer(24);
        FieldPosition pos = new FieldPosition(NumberFormat.INTEGER_FIELD);

        BigDecimal bd1 = new BigDecimal(22.3423D);
        dest1 = numForm.format(bd1, dest1, pos);
        System.out.println("dest1 = " + dest1);
        System.out.println("INTEGER portion is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());

        BigDecimal bd2 = new BigDecimal(64000D);
        dest2 = numForm.format(bd2, dest2, pos);
        System.out.println("dest2 = " + dest2);
        System.out.println("INTEGER portion is at: " + pos.getBeginIndex() +
            ", " + pos.getEndIndex());
    }
```

```
}

/*
Output:
dest1 = 22.342
INTEGER portion is at: 0, 2
dest2 = 64,000
INTEGER portion is at: 0, 6
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getAvailableLocales Method

Gets an array of all Locale objects that are supported by the NumberFormat class.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Locale[] getAvailableLocales();
```

Return Value

An array of all the Locale objects that are available.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getCurrencyInstance Method

Creates an instance of a NumberFormat object that can be used to format and parse currency values.

Overload List

Name	Description
NumberFormat.getCurrencyInstance ()	Creates an instance of a NumberFormat object that can be used to format and parse currency values.
NumberFormat.getCurrencyInstance (Locale)	Creates an instance of a NumberFormat object that can be used to format and parse currency values for the given locale.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getCurrencyInstance Method ()

Creates an instance of a NumberFormat object that can be used to format and parse currency values.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.NumberFormat getCurrencyInstance();
```

Return Value

An instance of a NumberFormat object that has the pattern set to the one used to format and parse currency values.

Example

The following example gets a default instance of a NumberFormat object appropriate for storing currency values.

```
// numberformat_getcurrencyinstance.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getCurrencyInstance();

        // Format some decimals using the pattern supplied above.
        String dest1 = numForm.format(22.3423D);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000D);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = $22.34
dest2 = $64,000.00
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getCurrencyInstance Method (Locale)

Creates an instance of a NumberFormat object that can be used to format and parse currency values for the given locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.NumberFormat getCurrencyInstance(
    java.util.Locale loc);
```

Parameters

loc

A Locale object containing the locale that will be used to format and parse the currency value.

Return Value

An instance of a NumberFormat object that has the pattern set to the one used to format and parse currency values for the given locale.

Example

The following example gets a default instance of a NumberFormat object appropriate for storing currency values using the en_CA (Canada) locale.

```
// numberformat_getcurrencyinstance_2.jsl

import java.text.*;
import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance using the
        // en_CA locale.
        Locale caLoc = new Locale("en", "CA");
        NumberFormat numForm =
            NumberFormat.getCurrencyInstance(caLoc);

        // Format some decimals using the pattern supplied above.
        String dest1 = numForm.format(22.3423D);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000D);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = $22.34
dest2 = $64,000.00
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.hashCode Method

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

A hash code for the current object.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getInstance Method

Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.

Overload List

Name	Description
NumberFormat.getInstance ()	Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.
NumberFormat.getInstance (Locale)	Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers for the given locale.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getInstance Method ()

Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.NumberFormat getInstance();
```

Return Value

An instance of a NumberFormat object that has the pattern set to the default pattern for numbers.

Example

The following example gets a default instance of a NumberFormat object appropriate for storing numbers.

```
// numberformat_getinstance.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Format some decimals using the pattern supplied above.
        String dest1 = numForm.format(22.3423D);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000D);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 22.342
dest2 = 64,000
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getInstance Method (Locale)

Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers for the given locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.NumberFormat getInstance(
    java.util.Locale loc);
```

Parameters

loc

A Locale object containing the locale that will be used to format and parse numbers.

Return Value

An instance of a NumberFormat object that has the pattern set to the default pattern for numbers in the given locale.

Example

The following example gets a default instance of a NumberFormat object appropriate for storing numbers using the de_DE (Germany) locale.

```
// numberformat_getinstance_2.jsl

import java.text.*;
import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance using the
        // de_DE locale.
        Locale deLoc = new Locale("de", "DE");
        NumberFormat numForm = NumberFormat.getInstance(deLoc);

        // Format some decimals using the pattern supplied above.
        String dest1 = numForm.format(22.3423D);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000D);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 22,342
dest2 = 64.000
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getMaximumFractionDigits Method

Gets a value representing the maximum number of digits displayed in the fractional part of the number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getMaximumFractionDigits();
```

Return Value

A value representing the maximum number of digits displayed in the fractional part of the number.

Example

The following example displays the various properties of the NumberFormat object, including the maximum fraction digits.

```
// numberformat_getmaximumfractiondigits.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getMaximumIntegerDigits Method

Gets a value representing the maximum number of digits displayed in the integer part of the number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getMaximumIntegerDigits();
```

Return Value

A value representing the maximum number of digits displayed in the integer part of the number.

Example

The following example displays the various properties of the NumberFormat object, including the maximum integer digits.

```
// numberformat_getmaximumintegerdigits.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getMinimumFractionDigits Method

Gets a value representing the minimum number of digits displayed in the fractional part of the number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getMinimumFractionDigits();
```

Return Value

A value representing the minimum number of digits displayed in the fractional part of the number.

Example

The following example displays the various properties of the NumberFormat object, including the minimum fraction digits.

```
// numberformat_getminimumfractiondigits.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getMinimumIntegerDigits Method

Gets a value representing the minimum number of digits displayed in the integer part of the number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getMinimumIntegerDigits();
```

Return Value

A value representing the minimum number of digits displayed in the integer part of the number.

Example

The following example displays the various properties of the NumberFormat object, including the minimum integer digits.

```
// numberformat_getminimumintegerdigits.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getNumberInstance Method

Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.

Overload List

Name	Description
NumberFormat.getNumberInstance ()	Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.
NumberFormat.getNumberInstance (Locale)	Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers for the given locale.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getNumberInstance Method ()

Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.NumberFormat getNumberInstance();
```

Return Value

An instance of a NumberFormat object that has the pattern set to the default pattern for numbers.

Example

The following example gets a default instance of a NumberFormat object appropriate for storing numbers.

```
// numberformat_getnumberinstance.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm =
            NumberFormat.getNumberInstance();

        // Format some decimals using the pattern supplied above.
        String dest1 = numForm.format(22.3423D);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000D);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 22.342
dest2 = 64,000
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getNumberInstance Method (Locale)

Creates an instance of a NumberFormat object and sets the pattern to the default pattern for numbers for the given locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.NumberFormat getNumberInstance(
    java.util.Locale loc);
```

Parameters

loc

A Locale object containing the locale that will be used to format and parse numbers.

Return Value

An instance of a NumberFormat object that has the pattern set to the default pattern for numbers in the given locale.

Example

The following example gets a default instance of a NumberFormat object appropriate for storing numbers using the th_TH (Thailand) locale.

```
// numberformat_getnumberinstance_2.jsl

import java.text.*;
import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance using the
        // th_TH locale.
        Locale thLoc = new Locale("th", "TH");
        NumberFormat numForm =
            NumberFormat.getNumberInstance(thLoc);

        // Format some decimals using the pattern supplied above.
        String dest1 = numForm.format(22.3423D);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000D);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 22.342
dest2 = 64,000
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getPercentInstance Method

Creates an instance of a NumberFormat object and sets the pattern to the one used to format and parse percentage values.

Overload List

Name	Description
NumberFormat.getPercentInstance ()	Creates an instance of a NumberFormat object and sets the pattern to the one used to format and parse percentage values.
NumberFormat.getPercentInstance (Locale)	Creates an instance of a NumberFormat object and sets the pattern to the one used to format and parse percentage values in the given locale.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getPercentInstance Method ()

Creates an instance of a NumberFormat object and sets the pattern to the one used to format and parse percentage values.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static final java.text.NumberFormat getPercentInstance();
```

Return Value

An instance of a NumberFormat object that has the pattern set to the one used to format and parse percentage values.

Example

The following example gets a default instance of a NumberFormat object appropriate for storing percentages.

```
// numberformat_getpercentinstance.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm =
            NumberFormat.getPercentInstance();

        // Format some decimals using the pattern supplied above.
        String dest1 = numForm.format(22.3423D);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000D);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 2,234.23%
dest2 = 6,400,000%
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.getPercentInstance Method (Locale)

Creates an instance of a NumberFormat object and sets the pattern to the one used to format and parse percentage values in the given locale.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public static java.text.NumberFormat getPercentInstance(
    java.util.Locale loc);
```

Parameters

loc

A Locale object containing the locale that will be used to format and parse the percentage value.

Return Value

An instance of a NumberFormat object that has the pattern set to the one used to format and parse percentage values in the given locale.

Example

The following example gets a default instance of a NumberFormat object appropriate for storing percentages using the es_ES (Spain) locale.

```
// numberformat_getpercentinstance_2.jsl

import java.text.*;
import java.util.Locale;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance using the
        // es_ES locale.
        Locale esLoc = new Locale("es", "ES");
        NumberFormat numForm =
            NumberFormat.getPercentInstance(esLoc);

        // Format some decimals using the pattern supplied above.
        String dest1 = numForm.format(22.3423D);
        System.out.println("dest1 = " + dest1);

        String dest2 = numForm.format(64000D);
        System.out.println("dest2 = " + dest2);
    }
}

/*
Output:
dest1 = 2.234,23%
dest2 = 6.400.000%
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.isGroupingUsed Method

Determines whether digits are being grouped together.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean isGroupingUsed();
```

Return Value

true if digits are being grouped together; false otherwise.

Example

The following example displays the various properties of the NumberFormat object, including the flag that determines if grouping is used.

```
// numberformat_isgroupingused.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.isParseIntegerOnly Method

Determines whether only the integer portion of the number is being parsed or if the fractional part is also included.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean isParseIntegerOnly();
```

Return Value

true if only the integer portion of the number is being parsed; false otherwise.

Example

The following example displays the various properties of the NumberFormat object, including the flag that determines if only the integer portion is to be parsed.

```
// numberformat_isparseintegeronly.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.parse Method

Parses a string into a number.

Overload List

Name	Description
NumberFormat.parse (String)	Parses a string into a number represented by a Number object.
NumberFormat.parse (String, ParsePosition)	Parses a string into a number represented by a Number object.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.parse Method (String)

Parses a string into a number represented by a Number object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Number parse(
    java.lang.String sourceStr) throws java.text.ParseException;
```

Parameters

sourceStr

The string to be parsed into a number represented by a Number object.

Return Value

A Number object containing the number parsed from the string.

Example

The following example shows how to create a Number object using a string representation of a decimal.

```
// numberformat_parse.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        try
        {
            // Get a default NumberFormat instance.
            NumberFormat numForm = NumberFormat.getInstance();

            // Parse a decimal value.
            Number n = numForm.parse("123456789.14");
            System.out.println("Number = " + n);
        }
        catch (ParseException ex)
        {
            System.out.println(ex.toString());
        }
    }
}

/*
Output:
Number = 1.2345678914E8
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.parse Method (String, ParsePosition)

Parses a string into a number represented by a Number object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Number parse(
    java.lang.String sourceStr,
    java.text.ParsePosition pos);
```

Parameters

sourceStr

The string to be parsed into a number represented by a Number object.

pos

Used to indicate the first position in the string to be parsed.

Return Value

A Number object containing the number parsed from the string.

Example

The following example parses a string starting at position 8 into a number and displays the result to the Console.

```
// numberformat_parse_2.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Parse the decimal starting at position 8.
        ParsePosition pos = new ParsePosition(8);
        Number n = numForm.parse("123456789.14", pos);

        System.out.println("Number = " + n);
        System.out.println("Position after parse = " +
            pos.getIndex());
    }
}

/*
Output:
Number = 9.14
Position after parse = 12
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.parseObject Method

Parses a string into an Object.

Overload List

Name	Description
NumberFormat.parseObject (String)	Parses a string into an Object.
NumberFormat.parseObject (String, ParsePosition)	Parses a string into an Object.

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.parseObject Method (String, ParsePosition)

Parses a string into an Object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.Object parseObject(
    java.lang.String sourceStr,
    java.text.ParsePosition pos);
```

Parameters

sourceStr

The string to be parsed into an Object.

pos

Used to indicate the first position in the string to be parsed.

Return Value

An Object containing the number represented by the string.

Example

The following example parses a string starting at position 8 into an Object and displays the result to the Console.

```
// numberformat_parseobject.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Parse the decimal starting at position 8.
        ParsePosition pos = new ParsePosition(8);
        Object o = numForm.parseObject("123456789.14", pos);
        System.out.println("Number = " + o.toString());
    }
}

/*
Output:
Number = 9.14
*/
```

See Also

Reference

NumberFormat Class

Concepts

NumberFormat Members

java.text Package

NumberFormat.setGroupingUsed Method

Sets a value determining whether digits are being grouped together.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setGroupingUsed(
    boolean newValue);
```

Parameters

newValue

true to group the digits together; false otherwise.

Example

The following example sets the grouping flag to false.

```
// numberformat_setgroupingused.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Set the grouping flag to false.
        numForm.setGroupingUsed(false);

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? false
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class
Concepts
NumberFormat Members
java.text Package

NumberFormat.setMaximumFractionDigits Method

Sets the value representing the maximum number of digits displayed in the fractional part of the number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setMaximumFractionDigits(
    int maxFractionDigits);
```

Parameters

maxFractionDigits

The maximum number of digits to be displayed in the fractional part of the number.

Example

The following example sets the value for the maximum fraction digits to 10.

```
// numberformat_setmaximumfractiondigits.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Set the maximum fraction digits to 10.
        numForm.setMaximumFractionDigits(10);

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 10
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class
Concepts
NumberFormat Members
java.text Package

NumberFormat.setMaximumIntegerDigits Method

Sets the value representing the maximum number of digits displayed in the integer part of the number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setMaximumIntegerDigits(
    int maxIntegerDigits);
```

Parameters

maxIntegerDigits

The maximum number of digits to be displayed in the integer part of the number.

Example

The following example sets the value for the maximum integer digits to 10.

```
// numberformat_setmaximumintegerdigits.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Set the maximum integer digits to 10.
        numForm.setMaximumIntegerDigits(10);

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 10
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class
Concepts
NumberFormat Members
java.text Package

NumberFormat.setMinimumFractionDigits Method

Sets the value representing the minimum number of digits displayed in the fractional part of the number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setMinimumFractionDigits(
    int minFractionDigits);
```

Parameters

minFractionDigits

The minimum number of digits to be displayed in the fractional part of the number.

Example

The following example sets the value for the minimum fraction digits to 10.

```
// numberformat_setminimumfractiondigits.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Set the minimum fraction digits to 10.
        numForm.setMinimumFractionDigits(10);

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 10
Maximum Integer Digits: 127
Minimum Fraction Digits: 10
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class
Concepts
NumberFormat Members
java.text Package

NumberFormat.setMinimumIntegerDigits Method

Sets the value representing the minimum number of digits displayed in the integer part of the number.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setMinimumIntegerDigits(
    int minIntegerDigits);
```

Parameters

minIntegerDigits

The minimum number of digits to be displayed in the integer part of the number.

Example

The following example sets the value for the minimum integer digits to 10.

```
// numberformat_setminimumintegerdigits.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Set the minimum integer digits to 10.
        numForm.setMinimumIntegerDigits(10);

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 10
Is Grouping Used? true
Is Parse Integer Only? false
*/
```

See Also

Reference

NumberFormat Class
Concepts
NumberFormat Members
java.text Package

NumberFormat.setParseIntegerOnly Method

Sets a value determining whether only the integer portion of the number is being parsed or if the fractional part is also included.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setParseIntegerOnly(
    boolean value);
```

Parameters

value

true to parse only the integer portion of the number; false otherwise.

Example

The following example sets the parse integer only flag to true.

```
// numberformat_setparseintegeronly.jsl

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get a default NumberFormat instance.
        NumberFormat numForm = NumberFormat.getInstance();

        // Set the parse integer only flag to true.
        numForm.setParseIntegerOnly(true);

        // Display the various properties of the NumberFormat
        // class.
        System.out.println("Maximum Fraction Digits: " +
            numForm.getMaximumFractionDigits());
        System.out.println("Maximum Integer Digits: " +
            numForm.getMaximumIntegerDigits());
        System.out.println("Minimum Fraction Digits: " +
            numForm.getMinimumFractionDigits());
        System.out.println("Minimum Integer Digits: " +
            numForm.getMinimumIntegerDigits());
        System.out.println("Is Grouping Used? " +
            numForm.isGroupingUsed());
        System.out.println("Is Parse Integer Only? " +
            numForm.isParseIntegerOnly());

    }
}

/*
Output:
Maximum Fraction Digits: 3
Maximum Integer Digits: 127
Minimum Fraction Digits: 0
Minimum Integer Digits: 1
Is Grouping Used? true
Is Parse Integer Only? true
*/
```

See Also

Reference
NumberFormat Class
Concepts
NumberFormat Members
java.text Package

ParseException Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public class java.text.ParseException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.text.ParseException

See Also

Concepts

ParseException Members

java.text Package

ParseException Members

The following tables list the members exposed by the ParseException type.

Public Constructors

Name	Description
ParseException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
getErrorOffset	
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden.
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ParseException Class

Concepts

java.text Package

ParseException Constructor

Overload List

Name	Description
ParseException (SerializationInfo, StreamingContext)	
ParseException (String, Exception)	
ParseException (String, int)	
ParseException (String, int, Exception)	

See Also

Reference

ParseException Class

Concepts

ParseException Members

java.text Package

ParseException Constructor (SerializationInfo, StreamingContext)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
protected java.text.ParseException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also
Reference
ParseException Class
Concepts
ParseException Members
java.text Package

ParseException Constructor (String, Exception)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.ParseException(
    java.lang.String msg,
    System.Exception inner);
```

Parameters

msg

inner

See Also
Reference
ParseException Class
Concepts
ParseException Members
java.text Package

ParseException Constructor (String, Int32)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.ParseException(
    java.lang.String msg,
    int errorOffset);
```

Parameters

msg

errorOffset

See Also
Reference
ParseException Class
Concepts
ParseException Members
java.text Package

ParseException Constructor (String, Int32, Exception)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.ParseException(
    java.lang.String msg,
    int errorOffset,
    System.Exception inner);
```

Parameters

msg

errorOffset

inner

See Also

Reference

ParseException Class

Concepts

ParseException Members

java.text Package

ParseException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
getErrorOffset	
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden.
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ParseException Class

Concepts

java.text Package

ParseException.getErrorOffset Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getErrorOffset();
```

See Also

Reference

ParseException Class

Concepts

ParseException Members

java.text Package

ParseException.GetObjectData Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

ParseException Class

Concepts

ParseException Members

java.text Package

ParseException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ParseException Class

Concepts

java.text Package

ParsePosition Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public class java.text.ParsePosition
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.text.ParsePosition

See Also

Concepts

ParsePosition Members

java.text Package

ParsePosition Members

The following tables list the members exposed by the ParsePosition type.

Public Constructors

Name	Description
ParsePosition	

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getIndex	
getClass	(inherited from Object)
clone	
setIndex	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ParsePosition Class

Concepts

java.text Package

ParsePosition Constructor

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.ParsePosition(
    int index);
```

Parameters
index

See Also
Reference
ParsePosition Class
Concepts
ParsePosition Members
java.text Package

ParsePosition Methods

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getIndex	
getClass	(inherited from Object)
clone	
setIndex	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ParsePosition Class

Concepts

java.text Package

ParsePosition.getIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getIndex();
```

See Also

Reference

ParsePosition Class

Concepts

ParsePosition Members

java.text Package

ParsePosition.setIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setIndex(
    int index);
```

Parameters
index

See Also
Reference
ParsePosition Class
Concepts
ParsePosition Members
java.text Package

RuleBasedCollator Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public class java.text.RuleBasedCollator
    extends java.text.Collator
```

Inheritance Hierarchy

java.lang.Object

  java.text.Collator

   java.text.RuleBasedCollator

See Also

Concepts

RuleBasedCollator Members

java.text Package

RuleBasedCollator Members

The following tables list the members exposed by the RuleBasedCollator type.

Public Methods

Name	Description
compare	Overridden.
equals	Determines whether two strings are equal. (inherited from Collator)
equals	Overridden.
getCollationKey	Overridden.
getDecomposition	Gets the decomposition type being used for collation. (inherited from Collator)
hashCode	Overridden.
getStrength	Gets the strength type being used for collation. (inherited from Collator)
getClass	(inherited from Object)
clone	
setDecomposition	Sets the decomposition type being used for collation. (inherited from Collator)
setStrength	Sets the strength type being used for collation. (inherited from Collator)
toString	Displays a human-readable representation of a Collator object. (inherited from Collator)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

RuleBasedCollator Class

Concepts

java.text Package

RuleBasedCollator Methods

Public Methods

Name	Description
compare	Overridden.
equals	Determines whether two strings are equal. (inherited from Collator)
equals	Overridden.
getCollationKey	Overridden.
getDecomposition	Gets the decomposition type being used for collation. (inherited from Collator)
hashCode	Overridden.
getStrength	Gets the strength type being used for collation. (inherited from Collator)
getClass	(inherited from Object)
clone	
setDecomposition	Sets the decomposition type being used for collation. (inherited from Collator)
setStrength	Sets the strength type being used for collation. (inherited from Collator)
toString	Displays a human-readable representation of a Collator object. (inherited from Collator)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

RuleBasedCollator Class

Concepts

java.text Package

RuleBasedCollator.clone Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object clone();
```

See Also

Reference

RuleBasedCollator Class

Concepts

RuleBasedCollator Members

java.text Package

RuleBasedCollator.compare Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int compare(
    java.lang.String source,
    java.lang.String target);
```

Parameters

source

target

See Also

Reference

RuleBasedCollator Class

Concepts

RuleBasedCollator Members

java.text Package

RuleBasedCollator.equals Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

RuleBasedCollator Class

Concepts

RuleBasedCollator Members

java.text Package

RuleBasedCollator.getCollationKey Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.CollationKey getCollationKey(
    java.lang.String source);
```

Parameters

source

See Also

Reference

RuleBasedCollator Class

Concepts

RuleBasedCollator Members

java.text Package

RuleBasedCollator.hashCode Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

RuleBasedCollator Class

Concepts

RuleBasedCollator Members

java.text Package

SimpleDateFormat Class

Contains methods and properties used to format a Date object as a string and to parse a string into a Date object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public class java.text.SimpleDateFormat
    extends java.text.DateFormat
```

Example

The following example demonstrates some common uses of the SimpleDateFormat class, such as formatting a date, parsing a date, and setting a pattern.

```
// simpledateformat_overview.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a SimpleDateFormat object.
        String pattern = "dd-MMM-yyyy HH:mm:ss.SSS";
        SimpleDateFormat sdf = new SimpleDateFormat();
        sdf.applyPattern(pattern);

        // Parse a sample date starting at position 0.
        String sampleDateStr = "04-Jul-1776 22:00:00.000";
        ParsePosition pos = new ParsePosition(0);
        Date sampleDate = sdf.parse(sampleDateStr, pos);

        // Print out the date using the supplied pattern.
        System.out.println("The pattern being used is " +
            sdf.toPattern());
        System.out.println("A sample date looks like " +
            sdf.format(sampleDate));
    }
}

/*
Sample output:
The pattern being used is dd-MMM-yyyy HH:mm:ss.SSS
A sample date looks like 04-Jul-1776 22:00:00.000
*/
```

Inheritance Hierarchy

java.lang.Object

  java.text.Format

    java.text.DateFormat

      java.text.SimpleDateFormat

See Also

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat Members

Contains methods and properties used to format a Date object as a string and to parse a string into a Date object.

The following tables list the members exposed by the SimpleDateFormat type.

Public Constructors

Name	Description
SimpleDateFormat	Overloaded. Initializes a new instance of a SimpleDateFormat object.

Public Fields

Name	Description
calendar	The underlying Calendar object being used for dates and times.(inherited from DateFormat)
numberFormat	The NumberFormat object used to format and parse a date and a time.(inherited from DateFormat)

Public Methods

Name	Description
applyLocalizedPattern	Changes the pattern that an instance of a SimpleDateFormat object is using after it is created. Localization of the pattern is currently not supported by the J# Class Libraries.
applyPattern	Changes the pattern that an instance of a SimpleDateFormat object is using after it is created.
equals	Overridden. Determines whether two SimpleDateFormat objects are equal.
format	Overloaded. Formats a Date object into a string.
getCalendar	Gets the Calendar object that is being used. The calendar is used to determine the format of a date. (inherited from DateFormat)
getDateFormatSymbols	Gets the DateFormatSymbols object that contains the symbols used when formatting a Date object.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getNumberFormat	Gets the NumberFormat object used to format and parse a date and a time. (inherited from DateFormat)
getTimeZone	Gets the TimeZone object associated with the Calendar object of this instance. (inherited from DateFormat)
getClass	(inherited from Object)
isLenient	Determines whether parsing is set to lenient for this DateFormat object. Leniency is determined by the Calendar object associated with this instance. (inherited from DateFormat)
clone	Creates a shallow copy of the current SimpleDateFormat object.
parse	Overloaded. Parses a string into a Date object.
parseObject	Overloaded. Parses a string into an Object. (inherited from DateFormat)

setCalendar	Sets the Calendar object that is being used. The calendar is used to determine the format of a date. (inherited from DateFormat)
setDateFormatSymbols	Sets a DateFormatSymbols object that contains the symbols used when formatting a Date.
setLenient	Sets a value that determines whether parsing is set to lenient for this DateFormat object. Leniency is determined by the Calendar object associated with this instance. (inherited from DateFormat)
setNumberFormat	Sets the NumberFormat object used to format and parse a date and a time. (inherited from DateFormat)
setTimeZone	Sets a TimeZone object to be associated with the Calendar object of this instance. (inherited from DateFormat)
toLocalizedPattern	Applies the previously set pattern to the string for an instance of a SimpleDateFormat object. Localization of the pattern is currently not supported by the J# Class Libraries.
toPattern	Applies the previously set pattern to the string for an instance of a SimpleDateFormat object.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SimpleDateFormat Class

Concepts

java.text Package

SimpleDateFormat Fields

Public Fields

Name	Description
calendar	The underlying Calendar object being used for dates and times. (inherited from DateFormat)
numberFormat	The NumberFormat object used to format and parse a date and a time. (inherited from DateFormat)

See Also

Reference

SimpleDateFormat Class

Concepts

java.text Package

SimpleDateFormat Constructor

Initializes a new instance of a SimpleDateFormat object.

Overload List

Name	Description
SimpleDateFormat ()	Initializes a new instance of a SimpleDateFormat object.
SimpleDateFormat (String)	Initializes a new instance of a SimpleDateFormat object. The Date object will be formatted into a string using the pattern provided.
SimpleDateFormat (String, DateFormatSymbols)	Initializes a new instance of a SimpleDateFormat object. The Date object will be formatted into a string using the pattern and format symbols provided.
SimpleDateFormat (String, Locale)	Initializes a new instance of a SimpleDateFormat object. The Date object will be formatted into a string using the pattern and locale provided.

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat Constructor ()

Initializes a new instance of a SimpleDateFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.SimpleDateFormat();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat Constructor (String)

Initializes a new instance of a SimpleDateFormat object. The Date object will be formatted into a string using the pattern provided.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.SimpleDateFormat(
    java.lang.String pattern);
```

Parameters

pattern

The pattern describing how to format the Date object as a string.

Remarks

This class uses a pattern string to format or parse a date. The pattern string is a symbolic representation of various components of a date/time representation. For example, "dd-MM-yyyy" represents the dayOfMonth-Month-Year representation of a date.

The following table lists the valid contents for the pattern:

char	Description
d	Day of month
D	Day of year
F	Day of week in month
E	Day of week
w	Week of year
W	Week of month
M	Month
y	Year
G	Era
k	Hour of day (1 - 24)
H	Hour of day (0 - 23)
m	Minute
s	Second
S	Millisecond
a	AM/PM

h	Hour of day (1 - 12)
K	Hour of day (0 - 11)
z	Timezone
'	Single quote - delimiter for inserting strings into the formatted date

The number of placeholders used for the year indicates how many digits are to be used to format the year part of date. Also, the number of placeholders used for the month component indicates the month representation as follows:

"M" or "MM" - Format month as number (such as 8 or 08)

"MMM" - Format month as short name of month (such as Aug)

"MMMM" - Format month as long name of month (such as August)

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat Constructor (String, DateFormatSymbols)

Initializes a new instance of a SimpleDateFormat object. The Date object will be formatted into a string using the pattern and format symbols provided.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.SimpleDateFormat(
    java.lang.String pattern,
    java.text.DateFormatSymbols symbols);
```

Parameters

pattern

The pattern describing how to format the Date object as a string.

symbols

A DateFormatSymbols object describing the symbols to use in formatting the Date object as a string.

Remarks

This class uses a pattern string to format or parse a date. The pattern string is a symbolic representation of various components of a date/time representation. For example, "dd-MM-yyyy" represents the dayOfMonth-Month-Year representation of a date.

The following table lists the valid contents for the pattern:

char	Description
d	Day of month
D	Day of year
F	Day of week in month
E	Day of week
w	Week of year
W	Week of month
M	Month
y	Year
G	Era
k	Hour of day (1 - 24)
H	Hour of day (0 - 23)
m	Minute
s	Second

S	Millisecond
a	AM/PM
h	Hour of day (1 - 12)
K	Hour of day (0 - 11)
z	Timezone
'	Single quote - delimiter for inserting strings into the formatted date

The number of placeholders used for the year indicates how many digits are to be used to format the year part of date. Also, the number of placeholders used for the month component indicates the month representation as follows:

"M" or "MM" - Format month as number (such as 8 or 08)

"MMM" - Format month as short name of month (such as Aug)

"MMMM" - Format month as long name of month (such as August)

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat Constructor (String, Locale)

Initializes a new instance of a SimpleDateFormat object. The Date object will be formatted into a string using the pattern and locale provided.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.SimpleDateFormat(
    java.lang.String pattern,
    java.util.Locale loc);
```

Parameters

pattern

The pattern describing how to format the Date object as a string.

loc

A Locale object used to determine the format of the Date object when displayed as a string.

Remarks

This class uses a pattern string to format or parse a date. The pattern string is a symbolic representation of various components of a date/time representation. For example, "dd-MM-yyyy" represents the dayOfMonth-Month-Year representation of a date.

The following table lists the valid contents for the pattern:

char	Description
d	Day of month
D	Day of year
F	Day of week in month
E	Day of week
w	Week of year
W	Week of month
M	Month
y	Year
G	Era
k	Hour of day (1 - 24)
H	Hour of day (0 - 23)
m	Minute
s	Second

S	Millisecond
a	AM/PM
h	Hour of day (1 - 12)
K	Hour of day (0 - 11)
z	Timezone
'	Single quote - delimiter for inserting strings into the formatted date

The number of placeholders used for the year indicates how many digits are to be used to format the year part of date. Also, the number of placeholders used for the month component indicates the month representation as follows:

"M" or "MM" - Format month as number (such as 8 or 08)

"MMM" - Format month as short name of month (such as Aug)

"MMMM" - Format month as long name of month (such as August)

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat Methods

Public Methods

Name	Description
applyLocalizedPattern	Changes the pattern that an instance of a SimpleDateFormat object is using after it is created. Localization of the pattern is currently not supported by the J# Class Libraries.
applyPattern	Changes the pattern that an instance of a SimpleDateFormat object is using after it is created.
equals	Overridden. Determines whether two SimpleDateFormat objects are equal.
format	Overloaded. Formats a Date object into a string.
getCalendar	Gets the Calendar object that is being used. The calendar is used to determine the format of a date. (inherited from DateFormat)
getDateFormatSymbols	Gets the DateFormatSymbols object that contains the symbols used when formatting a Date object.
hashCode	Overridden. Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.
getNumberFormat	Gets the NumberFormat object used to format and parse a date and a time. (inherited from DateFormat)
getTimeZone	Gets the TimeZone object associated with the Calendar object of this instance. (inherited from DateFormat)
getClass	(inherited from Object)
isLenient	Determines whether parsing is set to lenient for this DateFormat object. Leniency is determined by the Calendar object associated with this instance. (inherited from DateFormat)
clone	Creates a shallow copy of the current SimpleDateFormat object.
parse	Overloaded. Parses a string into a Date object.
parseObject	Overloaded. Parses a string into an Object. (inherited from DateFormat)
setCalendar	Sets the Calendar object that is being used. The calendar is used to determine the format of a date. (inherited from DateFormat)
setDateFormatSymbols	Sets a DateFormatSymbols object that contains the symbols used when formatting a Date.
setLenient	Sets a value that determines whether parsing is set to lenient for this DateFormat object. Leniency is determined by the Calendar object associated with this instance. (inherited from DateFormat)
setNumberFormat	Sets the NumberFormat object used to format and parse a date and a time. (inherited from DateFormat)
setTimeZone	Sets a TimeZone object to be associated with the Calendar object of this instance. (inherited from DateFormat)

toLocalizedPattern	Applies the previously set pattern to the string for an instance of a SimpleDateFormat object. Localization of the pattern is currently not supported by the J# Class Libraries.
toPattern	Applies the previously set pattern to the string for an instance of a SimpleDateFormat object.
toString	Displays a human-readable representation of a Format object. (inherited from Format)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SimpleDateFormat Class

Concepts

java.text Package

SimpleDateFormat.applyLocalizedPattern Method

Changes the pattern that an instance of a SimpleDateFormat object is using after it is created. Localization of the pattern is currently not supported by the J# Class Libraries.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void applyLocalizedPattern(
    java.lang.String pattern);
```

Parameters

pattern

Specifies the style used to format to or parse from date strings.

Example

The following example shows how to apply a localized pattern to a SimpleDateFormat object.

```
// simpledateformat_applylocalizedpattern.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Apply a localized pattern to a SimpleDateFormat object.
        SimpleDateFormat sdf = new SimpleDateFormat();
        String pattern =
            "'This moment: 'dd MMM yyyy '@'hh 'hours and 'mm 'minutes.'";
        sdf.applyLocalizedPattern(pattern);

        // Print out the time now.
        System.out.println(sdf.format(new Date()));
    }
}

/*
Sample Output:
This moment: 11 Nov 2004 @10 hours and 16 minutes.
*/
```

Remarks

Localization of the pattern is currently not supported by the J# Class Libraries. Calling this method has the same effect as calling applyPattern.

This class uses a pattern string to format or parse a date. The pattern string is a symbolic representation of various components of a date/time representation. For example, "dd-MM-yyyy" represents the dayOfMonth-Month-Year representation of a date.

The following table lists the valid contents for the pattern:

char	Description
d	Day of month
D	Day of year

F	Day of week in month
E	Day of week
w	Week of year
W	Week of month
M	Month
y	Year
G	Era
k	Hour of day (1 - 24)
H	Hour of day (0 - 23)
m	Minute
s	Second
S	Millisecond
a	AM/PM
h	Hour of day (1 - 12)
K	Hour of day (0 - 11)
z	Timezone
'	Single quote - delimiter for inserting strings into the formatted date

The number of placeholders used for the year indicates how many digits are to be used to format the year part of date. Also, the number of placeholders used for the month component indicates the month representation as follows:

"M" or "MM" - Format month as number (such as 8 or 08)

"MMM" - Format month as short name of month (such as Aug)

"MMMM" - Format month as long name of month (such as August)

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.applyPattern Method

Changes the pattern that an instance of a SimpleDateFormat object is using after it is created.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void applyPattern(
    java.lang.String pattern);
```

Parameters

pattern

Specifies the style used to format to or parse from date strings.

Example

The following example shows how to apply a pattern to a SimpleDateFormat object.

```
// simpledateformat_applypattern.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Apply a pattern to a SimpleDateFormat object.
        SimpleDateFormat sdf = new SimpleDateFormat();
        String pattern =
            "'This moment: 'dd MMM yyyy '@'hh 'hours and 'mm 'minutes.'";
        sdf.applyPattern(pattern);

        // Print out the time now.
        System.out.println(sdf.format(new Date()));
    }
}

/*
Sample Output:
This moment: 11 Nov 2004 @10 hours and 16 minutes.
*/
```

Remarks

This class uses a pattern string to format or parse a date. The pattern string is a symbolic representation of various components of a date/time representation. For example, "dd-MM-yyyy" represents the dayOfMonth-Month-Year representation of a date.

The following table lists the valid contents for the pattern:

char	Description
d	Day of month
D	Day of year
F	Day of week in month

E	Day of week
w	Week of year
W	Week of month
M	Month
y	Year
G	Era
k	Hour of day (1 - 24)
H	Hour of day (0 - 23)
m	Minute
s	Second
S	Millisecond
a	AM/PM
h	Hour of day (1 - 12)
K	Hour of day (0 - 11)
z	Timezone
'	Single quote - delimiter for inserting strings into the formatted date

The number of placeholders used for the year indicates how many digits are to be used to format the year part of date. Also, the number of placeholders used for the month component indicates the month representation as follows:

"M" or "MM" - Format month as number (such as 8 or 08)

"MMM" - Format month as short name of month (such as Aug)

"MMMM" - Format month as long name of month (such as August)

See Also
Reference
SimpleDateFormat Class
Concepts
SimpleDateFormat Members
java.text Package

SimpleDateFormat.clone Method

Creates a shallow copy of the current SimpleDateFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object clone();
```

Return Value

A shallow copy of the current SimpleDateFormat object.

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.equals Method

Determines whether two SimpleDateFormat objects are equal.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

A SimpleDateFormat object to compare with the current SimpleDateFormat object for equality.

Return Value

true if the two SimpleDateFormat objects are equal; false otherwise.

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.format Method

Formats a Date object into a string.

Overload List

Name	Description
SimpleDateFormat.format (Date)	Formats a Date object into a string.
SimpleDateFormat.format (Object)	Formats an object into a string.
SimpleDateFormat.format (Date, StringBuffer, FieldPosition)	Formats a Date object and appends it to a StringBuffer object.
SimpleDateFormat.format (Object, StringBuffer, FieldPosition)	Formats an object representing a Date and appends it to a StringBuffer object.

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.format Method (Date, StringBuffer, FieldPosition)

Formats a Date object and appends it to a StringBuffer object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.StringBuffer format(
    java.util.Date date,
    java.lang.StringBuffer appendBuf,
    java.text.FieldPosition pos);
```

Parameters

date

The Date object to be formatted into a string.

appendBuf

A StringBuffer object to be appended with the Date object formatted into a string.

pos

Used to find start and end indices of a particular field of the date in the formatted string.

Return Value

A StringBuffer object appended with the formatted Date object.

Example

The following example shows how to format a SimpleDateFormat object and to determine where the year field can be found in the formatted date.

```
// simpledateformat_format.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Apply a pattern to a SimpleDateFormat object.
        String pattern =
            "'This moment: 'dd MMM yyyy '@'hh 'hours and 'mm 'minutes.'";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Print out the time now using the format method.
        StringBuffer buf = new StringBuffer();
        FieldPosition pos = new FieldPosition(DateFormat.YEAR_FIELD);
        sdf.format(new Date(), buf, pos);
        System.out.println(buf);
        System.out.println("The year field occupies indices: " +
            pos.getBeginIndex() + " - " + pos.getEndIndex());
    }
}

/*
Sample Output:
This moment: 11 Nov 2004 @10 hours and 29 minutes.
The year field occupies indices: 20 - 24
*/
```

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.getDateFormatSymbols Method

Gets the DateFormatSymbols object that contains the symbols used when formatting a Date object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.DateFormatSymbols getDateFormatSymbols();
```

Return Value

The DateFormatSymbols object that contains the symbols used when formatting a Date object.

Example

The following example shows how to get the date format symbols.

```
// simpledateformat_getdateformatsymbols.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Apply a pattern to a SimpleDateFormat object.
        String pattern =
            "'This moment: 'dd MMM yyyy '@'hh 'hours and 'mm 'minutes.'";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Get the date format symbols for this SimpleDateFormat
        // object.
        DateFormatSymbols dfs = sdf.getDateFormatSymbols();

        // Print out the valid months in short name format.
        String[] shortMonths = dfs.getShortMonths();
        System.out.print("Months: ");
        for (int i = 0; i < shortMonths.length; i++)
        {
            System.out.print(shortMonths[i] + " ");
        }
        System.out.println();

        // Print out the valid eras.
        String[] eras = dfs.getEras();
        System.out.print("Eras: ");
        for (int i = 0; i < eras.length; i++)
        {
            System.out.print(eras[i] + " ");
        }
        System.out.println();

        // Print out the valid weekdays.
        String[] weekdays = dfs.getWeekdays();
        System.out.print("Weekdays: ");
        for (int i = 0; i < weekdays.length; i++)
        {
            System.out.print(weekdays[i] + " ");
        }
        System.out.println();
    }
}
```

```
/*
Sample Output:
Months: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Eras: ?? A.D.
Weekdays:  Sunday Monday Tuesday Wednesday Thursday Friday Saturday
*/
```

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.hashCode Method

Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

A hash code for the current object.

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.parse Method

Parses a string into a Date object.

Overload List

Name	Description
SimpleDateFormat.parse (String)	Parses a string into a Date object.
SimpleDateFormat.parse (String, ParsePosition)	Parses a string into a Date object.

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.parse Method (String, ParsePosition)

Parses a string into a Date object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.util.Date parse(
    java.lang.String sourceStr,
    java.text.ParsePosition pos);
```

Parameters

sourceStr

The string to be parsed into a Date object.

pos

Used to indicate the first position in the string to be parsed.

Return Value

A Date object containing the date represented by the string.

Example

The following example shows how to parse a date that adheres to a previously defined pattern.

```
// simpledateformat_parse.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a SimpleDateFormat object.
        String pattern = "dd-MMM-yyyy HH:mm:ss.SSS";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Parse a sample date starting at position 0.
        String sampleDateStr = "04-Jul-1776 22:00:00.000";
        ParsePosition pos = new ParsePosition(0);
        Date sampleDate = sdf.parse(sampleDateStr, pos);

        // Print out the date using the supplied pattern.
        System.out.println(sdf.format(sampleDate));
    }
}

/*
Sample Output:
04-Jul-1776 22:00:00.000
*/
```

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.setDateFormatSymbols Method

Sets a DateFormatSymbols object that contains the symbols used when formatting a Date.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public void setDateFormatSymbols(
    java.text.DateFormatSymbols symbols);
```

Parameters

symbols

A DateFormatSymbols object that contains the symbols used when formatting a Date.

Example

The following example shows how to set a modified DateFormatSymbols object.

```
// simpledateformat_setdateformatsymbols.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Apply a pattern to a SimpleDateFormat object.
        String pattern =
            "'This moment: 'dd MMM yyyy '@'hh 'hours and 'mm 'minutes.'";
        SimpleDateFormat sdf = new SimpleDateFormat(pattern);

        // Create a new DateFormatSymbols with a few modifications
        // from the norm.
        DateFormatSymbols dfs = new DateFormatSymbols();
        String[] months = { "jan", "feb", "mar", "apr", "may",
                            "jun", "jul", "aug", "sep", "oct",
                            "nov", "dec" };
        dfs.setShortMonths(months);

        String[] eras = { "bc", "ad" };
        dfs.setEras(eras);

        String[] weekdays = { "montag", "dienstag", "mittwoch",
                              "donnerstag", "freitag", "samstag",
                              "sonntag" };
        dfs.setWeekdays(weekdays);

        sdf.setDateFormatSymbols(dfs);

        // Print out the valid months in short name format.
        String[] newMonths = dfs.getShortMonths();
        System.out.print("Months: ");
        for (int i = 0; i < newMonths.length; i++)
        {
            System.out.print(newMonths[i] + " ");
        }
        System.out.println();

        // Print out the valid eras.
        String[] newEras = dfs.getEras();
        System.out.print("Eras: ");
        for (int i = 0; i < newEras.length; i++)
```

```
        {
            System.out.print(newEras[i] + " ");
        }
        System.out.println();

        // Print out the valid weekdays.
        String[] newWeekdays = dfs.getWeekdays();
        System.out.print("Weekdays: ");
        for (int i = 0; i < newWeekdays.length; i++)
        {
            System.out.print(newWeekdays[i] + " ");
        }
        System.out.println();
    }
}

/*
Sample Output:
Months: jan feb mar apr may jun jul aug sep oct nov dec
Eras: bc ad
Weekdays: montag dienstag mittwoch donnerstag freitag samstag sonntag
*/
```

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.toLocalizedPattern Method

Applies the previously set pattern to the string for an instance of a SimpleDateFormat object. Localization of the pattern is currently not supported by the J# Class Libraries.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toLocalizedPattern();
```

Return Value

The string representation of the date formatted using the previously set pattern.

Example

The following example prints out the default localized pattern.

```
// simpledateformat_tolocalizedpattern.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a SimpleDateFormat object with the
        // default pattern.
        SimpleDateFormat sdf = new SimpleDateFormat();

        // Get the localized pattern.
        String locPattern = sdf.toLocalizedPattern();

        // Print out the localized pattern.
        System.out.println("Localized pattern = " + locPattern);
    }
}

/*
Sample Output:
Localized pattern = dd-MMM-yy hh:mm:ss a
*/
```

Remarks

Localization of the pattern is currently not supported by the J# Class Libraries. Calling this method has the same effect as calling toPattern.

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

SimpleDateFormat.toPattern Method

Applies the previously set pattern to the string for an instance of a SimpleDateFormat object.

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toPattern();
```

Return Value

The string representation of the date formatted using the previously set pattern.

Example

The following example prints out the default pattern.

```
// simpledateformat_topattern.jsl

import java.text.*;
import java.util.Date;

public class Program
{
    public static void main(String[] args)
    {
        // Create a SimpleDateFormat object with the
        // default pattern.
        SimpleDateFormat sdf = new SimpleDateFormat();

        // Get the pattern.
        String pattern = sdf.toPattern();

        // Print out the pattern.
        System.out.println("Default pattern = " + pattern);
    }
}

/*
Sample Output:
Default pattern = dd-MMM-yy hh:mm:ss a
*/
```

See Also

Reference

SimpleDateFormat Class

Concepts

SimpleDateFormat Members

java.text Package

StringCharacterIterator Class

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public final class java.text.StringCharacterIterator
    extends java.lang.Object
    implements java.text.CharacterIterator
```

Inheritance Hierarchy

java.lang.Object

java.text.StringCharacterIterator

See Also

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator Members

The following tables list the members exposed by the StringCharacterIterator type.

Public Constructors

Name	Description
StringCharacterIterator	Overloaded.

Public Methods

Name	Description
current	
equals	Overridden.
first	
getBeginIndex	
getEndIndex	
hashCode	Overridden.
getIndex	
getClass	(inherited from Object)
last	
clone	
next	
previous	
setIndex	
toString	Overridden.

See Also

Reference

StringCharacterIterator Class

Concepts

java.text Package

StringCharacterIterator Constructor

Overload List

Name	Description
StringCharacterIterator (String)	
StringCharacterIterator (String, int)	
StringCharacterIterator (String, int, int, int)	

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator Constructor (String)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.StringCharacterIterator(
    java.lang.String text);
```

Parameters

text

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator Constructor (String, Int32)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.StringCharacterIterator(
    java.lang.String text,
    int pos);
```

Parameters

text

pos

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator Constructor (String, Int32, Int32, Int32)

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public java.text.StringCharacterIterator(
    java.lang.String text,
    int begin,
    int end,
    int pos);
```

Parameters

text

begin

end

pos

See Also
Reference
StringCharacterIterator Class
Concepts
StringCharacterIterator Members
java.text Package

StringCharacterIterator Methods

Public Methods

Name	Description
current	
equals	Overridden.
first	
getBeginIndex	
getEndIndex	
hashCode	Overridden.
getIndex	
getClass	(inherited from Object)
last	
clone	
next	
previous	
setIndex	
toString	Overridden.

See Also

Reference

StringCharacterIterator Class

Concepts

java.text Package

StringCharacterIterator.clone Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.current Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char current();
```

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.equals Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.first Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char first();
```

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.getBeginIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getBeginIndex();
```

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.getEndIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getEndIndex();
```

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.hashCode Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.getIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public int getIndex();
```

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.last Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char last();
```

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.next Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char next();
```

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.previous Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char previous();
```

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

StringCharacterIterator.setIndex Method

Package: java.text

Assembly: vjslib (in vjslib.dll)

```
public char setIndex(
    int idx);
```

Parameters

idx

See Also

Reference

StringCharacterIterator Class

Concepts

StringCharacterIterator Members

java.text Package

java.util

Contains the classes and interfaces used to handle collections such as hash tables, linked lists, stacks, and dictionaries. This package also provides classes to handle time zones and globalization, in addition to other utility classes such as a random-number generator and a string tokenizer.

Classes

Class	Description
ArrayList	Represents a dynamically sized, index-based collection of objects.
Arrays	A class that contains several methods for sorting, searching, filling and comparing arrays.
Calendar	Contains methods and members to perform basic operations involving days, weeks, months, and years.
ConcurrentModificationException	The exception that is thrown when two threads attempt to modify a collection concurrently.
Date	Contains methods to perform basic operations on dates and times.
Dictionary	An abstract class that contains a map of elements associated with keys. Both elements and keys are non-null objects.
EmptyStackException	The exception that is thrown when attempting to pop an element off of an empty stack.
GregorianCalendar	Extends the abstract class Calendar and provides the common standard calendar.
HashMap	Represents a unique collection of key-value pairs. The elements are sorted according to the hash value of the key, and each key can exist only once in the collection.
HashSet	Represents a unique collection of elements that are sorted according to their hash value, and can each exist only once in the collection.
Hashtable	Provides an object for key entries mapped to their corresponding values. Both keys and values must be non-null objects.
LinkedList	A class that represents a dynamic collection of elements. Each element consists of a data field and a link field. The link field points to the data field of the next element.
MissingResourceException	An exception thrown to indicate that a resource is missing.
NoSuchElementException	The exception that is thrown when attempting to access an element in a collection that does not exist. This exception is frequently thrown when attempting to perform an operation on an empty collection or when attempting to access past the end of the collection.
Observable	Represents the observable object, which usually contains data. It can have one or more observers. An Observer object is an instance of a class that implements the Observer interface. When the data on the observable object changes, the observers can be notified through the update method, which passes the necessary information to the notifyObservers method.
Random	Contains methods to obtain various sorts of random data, such as integers, floats, and other primitives.

SimpleTimeZone	Represents a specialization of the TimeZone class containing methods useful for calculations involving time zones, such as offsets from UTC and determining whether daylight savings ti me is in effect.
Stack	Represents a collection of objects that are accessed in a last-in, first out fashion. Elements ar e pushed onto the top of the stack, and the last element pushed on is the first element to be popped off.
StringTokenizer	Provides methods to parse a string and split it into tokens based on a delimiter.
TimeZone	Represents an abstract class containing methods useful for calculations involving time zones , such as offsets from UTC and determining whether daylight savings time is in effect.
TreeMap	Provides a sorted map object that implements the SortedMap interface.
TreeSet	A class that represents a sorted set of elements. The elements are sorted using the comparat or provided through constructor or using the Comparable interface methods implemented b y the elements.
Vector	Represents a dynamic array of elements. Unlike a static array, a vector will grow in size once it has reached capacity.

Interfaces

Interface	Description
Collection	An interface that represents a group of objects or elements. It is implemented by many collection classes such as LinkedList, ArrayList, HashSet and TreeSet.
Comparator	Contains methods to compare objects. It is implemented by the abstract class Collator.
Enumeration	Provides an efficient mechanism to traverse the elements of various collections.
Iterator	Provides an efficient mechanism to traverse the elements of various collections.
List	An interface that represents an ordered collection of elements.
ListIterator	An interface that implements the Iterator interface. It allows traversing the list and modifying it during iteration. It also obtains the current position of the iterator's cursor. The cursor position is between the elements of a collecti on.
Map	An interface that provides a Map object for key entries and their corresponding values.
Observer	An interface that enables the implementing class to be notified of any changes in the observable objects.
Set	Provides a set of objects that implements the Collection interface.
SortedMap	Represents a specialization of a Map object containing elements sorted by their key.
SortedSet	Represents a specialization of a Set object containing sorted elements.

AbstractCollection Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.util.AbstractCollection
    extends java.lang.Object
    implements java.util.Collection
```

Inheritance Hierarchy

java.lang.Object

java.util.AbstractCollection

java.util.AbstractList

java.util.AbstractSet

See Also

Concepts

AbstractCollection Members

java.util Package

AbstractCollection Members

The following tables list the members exposed by the AbstractCollection type.

Public Constructors

Name	Description
AbstractCollection	

Public Methods

Name	Description
add	
addAll	
clear	
contains	
containsAll	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
isEmpty	
iterator	
clone	
remove	
removeAll	
retainAll	
size	
toArray	Overloaded.
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractCollection Class

Concepts

java.util Package

AbstractCollection Constructor

Initializes a new instance of the AbstractCollection Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.AbstractCollection();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection Methods

Public Methods

Name	Description
add	
addAll	
clear	
contains	
containsAll	
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
isEmpty	
iterator	
clone	
remove	
removeAll	
retainAll	
size	
toArray	Overloaded.
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractCollection Class

Concepts

java.util Package

AbstractCollection.add Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean add(
    java.lang.Object e);
```

Parameters

e

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.addAll Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean addAll(
    java.util.Collection c);
```

Parameters

c

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.clear Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.contains Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean contains(
    java.lang.Object e);
```

Parameters

e

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.containsAll Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean containsAll(
    java.util.Collection c);
```

Parameters

c

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.isEmpty Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isEmpty();
```

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.iterator Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Iterator iterator();
```

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.remove Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean remove(
    java.lang.Object e);
```

Parameters

e

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.removeAll Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean removeAll(
    java.util.Collection c);
```

Parameters

c

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.retainAll Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean retainAll(
    java.util.Collection c);
```

Parameters

c

See Also
Reference
AbstractCollection Class
Concepts
AbstractCollection Members
java.util Package

AbstractCollection.size Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int size();
```

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.toArray Method

Overload List

Name	Description
AbstractCollection.toArray ()	
AbstractCollection.toArray (Object[])	

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.toArray Method ()

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object[] toArray();
```

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.toArray Method (Object[ ])

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object[] toArray(
    java.lang.Object[] arr);
```

Parameters

arr

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractCollection.toString Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

AbstractCollection Class

Concepts

AbstractCollection Members

java.util Package

AbstractList Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.util.AbstractList
    extends java.util.AbstractCollection
    implements java.util.List
```

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractCollection

  java.util.AbstractList

    java.util.AbstractSequentialList

    java.util.ArrayList

    java.util.Vector

See Also

Concepts

AbstractList Members

java.util Package

AbstractList Members

The following tables list the members exposed by the AbstractList type.

Public Constructors

Name	Description
AbstractList	

Public Fields

Name	Description
modCount	

Public Methods

Name	Description
add	Overloaded. Overridden.
addAll	Overloaded. Overridden.
clear	Overridden.
contains	(inherited from AbstractCollection)
containsAll	(inherited from AbstractCollection)
equals	Overridden.
get	
hashCode	Overridden.
getClass	(inherited from Object)
indexOf	
isEmpty	(inherited from AbstractCollection)
iterator	Overridden.
lastIndexOf	
listIterator	Overloaded.
clone	(inherited from AbstractCollection)
remove	Overloaded.
removeAll	(inherited from AbstractCollection)
removeRange	

retainAll	(inherited from AbstractCollection)
set	
size	Overridden.
subList	
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractList Class

Concepts

java.util Package

AbstractList Fields

Public Fields

Name	Description
modCount	

See Also

Reference

AbstractList Class

Concepts

java.util Package

AbstractList.modCount Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected transient int modCount;
```

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList Constructor

Initializes a new instance of the AbstractList Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.AbstractList();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList Methods

Public Methods

Name	Description
add	Overloaded. Overridden.
addAll	Overloaded. Overridden.
clear	Overridden.
contains	(inherited from AbstractCollection)
containsAll	(inherited from AbstractCollection)
equals	Overridden.
get	
hashCode	Overridden.
getClass	(inherited from Object)
indexOf	
isEmpty	(inherited from AbstractCollection)
iterator	Overridden.
lastIndexOf	
listIterator	Overloaded.
clone	(inherited from AbstractCollection)
remove	Overloaded.
removeAll	(inherited from AbstractCollection)
removeRange	
retainAll	(inherited from AbstractCollection)
set	
size	Overridden.
subList	
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractList Class

Concepts

java.util Package

AbstractList.add Method

Overload List

Name	Description
AbstractList.add (Object)	
AbstractList.add (int, Object)	

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.add Method (Object)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean add(
    java.lang.Object e);
```

Parameters

e

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.add Method (Int32, Object)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void add(
    int ix,
    java.lang.Object e);
```

Parameters

ix

e

See Also
Reference
AbstractList Class
Concepts
AbstractList Members
java.util Package

AbstractList.addAll Method

Overload List

Name	Description
AbstractList.addAll (Collection)	
AbstractList.addAll (int, Collection)	

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.addAll Method (Collection)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean addAll(
    java.util.Collection c);
```

Parameters

c

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.addAll Method (Int32, Collection)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean addAll(
    int ix,
    java.util.Collection c);
```

Parameters

ix

c

See Also
Reference
AbstractList Class
Concepts
AbstractList Members
java.util Package

AbstractList.clear Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.equals Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object e);
```

Parameters

e

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.get Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object get(
    int ix);
```

Parameters

ix

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.hashCode Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.indexOf Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int indexOf(
    java.lang.Object e);
```

Parameters

e

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.iterator Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Iterator iterator();
```

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.lastIndexOf Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int lastIndexOf(
    java.lang.Object e);
```

Parameters

e

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.listIterator Method

Overload List

Name	Description
AbstractList.listIterator ()	
AbstractList.listIterator (int)	

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.listIterator Method ()

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ListIterator listIterator();
```

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.listIterator Method (Int32)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ListIterator listIterator(
    int ix);
```

Parameters

ix

See Also
Reference
AbstractList Class
Concepts
AbstractList Members
java.util Package

AbstractList.remove Method

Overload List

Name	Description
AbstractList.remove (int)	
AbstractList.remove (Object)	

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.remove Method (Int32)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object remove(
    int ix);
```

Parameters

ix

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.removeRange Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected void removeRange(
    int fromIx,
    int toIx);
```

Parameters

fromIx

toIx

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.set Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object set(
    int is,
    java.lang.Object e);
```

Parameters

is

e

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.size Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int size();
```

See Also

Reference

AbstractList Class

Concepts

AbstractList Members

java.util Package

AbstractList.subList Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.List subList(
    int fromIx,
    int toIx);
```

Parameters

fromIx

toIx

See Also
Reference
AbstractList Class
Concepts
AbstractList Members
java.util Package

AbstractMap Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.util.AbstractMap
    extends java.lang.Object
    implements java.util.Map
```

Inheritance Hierarchy

java.lang.Object

java.util.AbstractMap

java.util.HashMap

java.util.TreeMap

java.util.WeakHashMap

See Also

Concepts

AbstractMap Members

java.util Package

AbstractMap Members

The following tables list the members exposed by the AbstractMap type.

Public Constructors

Name	Description
AbstractMap	

Public Methods

Name	Description
clear	
containsKey	
containsValue	
entrySet	
equals	Overridden.
get	
hashCode	Overridden.
getClass	(inherited from Object)
isEmpty	
keySet	
clone	
put	
putAll	
remove	
size	
toString	Overridden.
values	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractMap Class

Concepts

java.util Package

AbstractMap Constructor

Initializes a new instance of the AbstractMap Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.AbstractMap();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap Methods

Public Methods

Name	Description
clear	
containsKey	
containsValue	
entrySet	
equals	Overridden.
get	
hashCode	Overridden.
getClass	(inherited from Object)
isEmpty	
keySet	
clone	
put	
putAll	
remove	
size	
toString	Overridden.
values	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractMap Class

Concepts

java.util Package

AbstractMap.clear Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.containsKey Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean containsKey(
    java.lang.Object k);
```

Parameters
k

See Also
Reference
AbstractMap Class
Concepts
AbstractMap Members
java.util Package

AbstractMap.containsValue Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean containsValue(
    java.lang.Object v);
```

Parameters

v

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.entrySet Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Set entrySet();
```

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.equals Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object c);
```

Parameters

c

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.get Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object get(
    java.lang.Object k);
```

Parameters
k

See Also
Reference
AbstractMap Class
Concepts
AbstractMap Members
java.util Package

AbstractMap.hashCode Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.isEmpty Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isEmpty();
```

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.keySet Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Set keySet();
```

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.put Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object put(
    java.lang.Object k,
    java.lang.Object v);
```

Parameters

k

v

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.putAll Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void putAll(
    java.util.Map m);
```

Parameters

m

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.remove Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object remove(
    java.lang.Object k);
```

Parameters
k

See Also
Reference
AbstractMap Class
Concepts
AbstractMap Members
java.util Package

AbstractMap.size Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.toString Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractMap.values Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Collection values();
```

See Also

Reference

AbstractMap Class

Concepts

AbstractMap Members

java.util Package

AbstractSequentialList Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.util.AbstractSequentialList
    extends java.util.AbstractList
```

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractCollection

    java.util.AbstractList

      java.util.AbstractSequentialList

        java.util.LinkedList

See Also

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList Members

The following tables list the members exposed by the AbstractSequentialList type.

Public Constructors

Name	Description
AbstractSequentialList	

Public Fields

Name	Description
modCount	(inherited from AbstractList)

Public Methods

Name	Description
add	Overloaded.
addAll	Overloaded.
clear	(inherited from AbstractList)
contains	(inherited from AbstractCollection)
containsAll	(inherited from AbstractCollection)
equals	(inherited from AbstractList)
get	Overridden.
hashCode	(inherited from AbstractList)
getClass	(inherited from Object)
indexOf	(inherited from AbstractList)
isEmpty	(inherited from AbstractCollection)
iterator	Overridden.
lastIndexOf	(inherited from AbstractList)
listIterator	Overloaded.
clone	(inherited from AbstractCollection)
remove	Overloaded. Overridden.
removeAll	(inherited from AbstractCollection)
removeRange	(inherited from AbstractList)

retainAll	(inherited from AbstractCollection)
set	Overridden.
size	(inherited from AbstractList)
subList	(inherited from AbstractList)
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractSequentialList Class

Concepts

java.util Package

AbstractSequentialList Fields

Public Fields

Name	Description
modCount	(inherited from AbstractList)

See Also

Reference

AbstractSequentialList Class

Concepts

java.util Package

AbstractSequentialList Constructor

Initializes a new instance of the AbstractSequentialList Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.AbstractSequentialList();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList Methods

Public Methods

Name	Description
add	Overloaded.
addAll	Overloaded.
clear	(inherited from AbstractList)
contains	(inherited from AbstractCollection)
containsAll	(inherited from AbstractCollection)
equals	(inherited from AbstractList)
get	Overridden.
hashCode	(inherited from AbstractList)
getClass	(inherited from Object)
indexOf	(inherited from AbstractList)
isEmpty	(inherited from AbstractCollection)
iterator	Overridden.
lastIndexOf	(inherited from AbstractList)
listIterator	Overloaded.
clone	(inherited from AbstractCollection)
remove	Overloaded. Overridden.
removeAll	(inherited from AbstractCollection)
removeRange	(inherited from AbstractList)
retainAll	(inherited from AbstractCollection)
set	Overridden.
size	(inherited from AbstractList)
subList	(inherited from AbstractList)
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractSequentialList Class

Concepts

java.util Package

AbstractSequentialList.add Method

Overload List

Name	Description
AbstractSequentialList.add (Object)	
AbstractSequentialList.add (int, Object)	

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList.add Method (Int32, Object)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void add(
    int ix,
    java.lang.Object e);
```

Parameters

ix

e

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList.addAll Method

Overload List

Name	Description
AbstractSequentialList.addAll (Collection)	
AbstractSequentialList.addAll (int, Collection)	

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList.addAll Method (Int32, Collection)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean addAll(
    int ix,
    java.util.Collection c);
```

Parameters

ix

c

See Also
Reference
AbstractSequentialList Class
Concepts
AbstractSequentialList Members
java.util Package

AbstractSequentialList.get Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object get(
    int ix);
```

Parameters

ix

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList.iterator Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Iterator iterator();
```

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList.listIterator Method

Overload List

Name	Description
AbstractSequentialList.listIterator ()	
AbstractSequentialList.listIterator (int)	

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList.listIterator Method (Int32)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.ListIterator listIterator(
    int ix);
```

Parameters

ix

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList.remove Method

Overload List

Name	Description
AbstractSequentialList.remove (int)	
AbstractSequentialList.remove (Object)	

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList.remove Method (Int32)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object remove(
    int ix);
```

Parameters

ix

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSequentialList.set Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object set(
    int ix,
    java.lang.Object e);
```

Parameters

ix

e

See Also

Reference

AbstractSequentialList Class

Concepts

AbstractSequentialList Members

java.util Package

AbstractSet Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.util.AbstractSet
    extends java.util.AbstractCollection
    implements java.util.Set
```

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractCollection

  java.util.AbstractSet

    java.util.HashSet

    java.util.TreeSet

See Also

Concepts

AbstractSet Members

java.util Package

AbstractSet Members

The following tables list the members exposed by the AbstractSet type.

Public Constructors

Name	Description
AbstractSet	

Public Methods

Name	Description
add	(inherited from AbstractCollection)
addAll	(inherited from AbstractCollection)
clear	(inherited from AbstractCollection)
contains	(inherited from AbstractCollection)
containsAll	(inherited from AbstractCollection)
equals	Overridden.
hashCode	Overridden.
getClass	(inherited from Object)
isEmpty	(inherited from AbstractCollection)
iterator	Overridden.
clone	(inherited from AbstractCollection)
remove	(inherited from AbstractCollection)
removeAll	(inherited from AbstractCollection)
retainAll	(inherited from AbstractCollection)
size	Overridden.
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractSet Class

Concepts

java.util Package

AbstractSet Constructor

Initializes a new instance of the AbstractSet Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.AbstractSet();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

AbstractSet Class

Concepts

AbstractSet Members

java.util Package

AbstractSet Methods

Public Methods

Name	Description
add	(inherited from AbstractCollection)
addAll	(inherited from AbstractCollection)
clear	(inherited from AbstractCollection)
contains	(inherited from AbstractCollection)
containsAll	(inherited from AbstractCollection)
equals	Overridden.
hashCode	Overridden.
getClass	(inherited from Object)
isEmpty	(inherited from AbstractCollection)
iterator	Overridden.
clone	(inherited from AbstractCollection)
remove	(inherited from AbstractCollection)
removeAll	(inherited from AbstractCollection)
retainAll	(inherited from AbstractCollection)
size	Overridden.
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

AbstractSet Class

Concepts

java.util Package

AbstractSet.equals Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object c);
```

Parameters

c

See Also

Reference

AbstractSet Class

Concepts

AbstractSet Members

java.util Package

AbstractSet.hashCode Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

AbstractSet Class

Concepts

AbstractSet Members

java.util Package

AbstractSet.iterator Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Iterator iterator();
```

See Also

Reference

AbstractSet Class

Concepts

AbstractSet Members

java.util Package

AbstractSet.size Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int size();
```

See Also

Reference

AbstractSet Class

Concepts

AbstractSet Members

java.util Package

ArrayList Class

Represents a dynamically sized, index-based collection of objects.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.ArrayList
    extends java.util.AbstractList
    implements java.util.List, java.lang.Cloneable, java.io.Serializable
```

Example

The following example illustrates the add, remove, size, clear, contains, get, set, and isEmpty methods of the ArrayList class.

```
// arraylist_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create an ArrayList with a capacity for 10 elements.
        ArrayList arrList = new ArrayList(10);

        // Add three elements to the ArrayList.
        arrList.add(new Integer(1));
        arrList.add(new Integer(2));
        arrList.add(new Integer(3));

        // Print the size of the ArrayList.
        int size = arrList.size();
        System.out.println("arrList contains " + size +
            " elements.");

        // Determine if the ArrayList contains an element.
        if (arrList.contains(new Integer(4)))
        {
            System.out.println("There is already an Integer with " +
                "the value 4 in the ArrayList.");
        }
        else
        {
            System.out.println("There is no Integer with " +
                "the value 4 in the ArrayList.");
        }

        // Change the value of the second element in the ArrayList.
        arrList.set(1, new Integer(4));

        // Print out the elements without using an iterator.
        for (int i = 0; i < arrList.size(); i++)
        {
            System.out.println(arrList.get(i));
        }

        // Remove the second element in the ArrayList.
        Object o = arrList.remove(1);
        System.out.println("Removed " + o.toString());

        // If the ArrayList is not empty, clear it out now.
        if (!arrList.isEmpty())
```

```
        {
            arrList.clear();
        }
    }
}

/*
Output:
arrList contains 3 elements.
There is no Integer with the value 4 in the ArrayList.
1
4
3
Removed 4
*/
```

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractCollection

    java.util.AbstractList

      java.util.ArrayList

See Also

Concepts

ArrayList Members

java.util Package

ArrayList Members

Represents a dynamically sized, index-based collection of objects.

The following tables list the members exposed by the ArrayList type.

Public Constructors

Name	Description
ArrayList	Overloaded. Creates and initializes a new instance of a ArrayList object.

Public Fields

Name	Description
modCount	(inherited from AbstractList)

Public Methods

Name	Description
add	Overloaded. Overridden. Adds an element to the end of a ArrayList.
addAll	Overloaded. Overridden. Adds all the items in an existing collection into a ArrayList object.
clear	Overridden. Removes all items from a ArrayList object.
contains	Overridden. Determines whether an element exists in a ArrayList object.
containsAll	(inherited from AbstractCollection)
ensureCapacity	Ensures that the length of a ArrayList object will be no smaller than the specified size.
equals	(inherited from AbstractList)
get	Overridden. Returns the element at the specified index.
hashCode	(inherited from AbstractList)
getClass	(inherited from Object)
indexOf	Overridden. Provides the index of an element in a ArrayList object.
isEmpty	Overridden. Determines whether a ArrayList object is empty.
iterator	(inherited from AbstractList)
lastIndexOf	Overridden. Provides the index of the last occurrence of an element in a ArrayList object.
listIterator	Overloaded. (inherited from AbstractList)
clone	Creates a new instance of a ArrayList object that is a shallow copy of an existing ArrayList object.
remove	Overloaded. Overridden. Removes an element from a ArrayList object.
removeAll	(inherited from AbstractCollection)

removeRange	Overridden. Removes all elements within the specified range from a ArrayList object.
retainAll	(inherited from AbstractCollection)
set	Overridden. Sets an element in a ArrayList object at the specified index.
size	Overridden. Provides the number of elements in a ArrayList object.
subList	(inherited from AbstractList)
toArray	Overloaded. Overridden. Creates a static array containing the elements of a ArrayList object.
toString	(inherited from AbstractCollection)
trimToSize	Trims a ArrayList object to the number of elements it contains.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ArrayList Class

Concepts

java.util Package

ArrayList Fields

Public Fields

Name	Description
modCount	(inherited from AbstractList)

See Also

Reference

ArrayList Class

Concepts

java.util Package

ArrayList Constructor

Creates and initializes a new instance of a ArrayList object.

Overload List

Name	Description
ArrayList ()	Constructs a ArrayList object.
ArrayList (Collection)	Constructs a ArrayList object, and copies all elements from the collection, passed as a parameter, to this ArrayList.
ArrayList (int)	Constructs a ArrayList object with the specified size.

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList Constructor ()

Constructs a ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ArrayList();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList Constructor (Collection)

Constructs a ArrayList object, and copies all elements from the collection, passed as a parameter, to this ArrayList.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ArrayList(
    java.util.Collection c);
```

Parameters

c

The name of a collection.

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList Constructor (Int32)

Constructs a ArrayList object with the specified size.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ArrayList(
    int initialCapacity);
```

Parameters

initialCapacity

The initial size of the list.

See Also
Reference
ArrayList Class
Concepts
ArrayList Members
java.util Package

ArrayList Methods

Public Methods

Name	Description
add	Overloaded. Overridden. Adds an element to the end of a ArrayList.
addAll	Overloaded. Overridden. Adds all the items in an existing collection into a ArrayList object.
clear	Overridden. Removes all items from a ArrayList object.
contains	Overridden. Determines whether an element exists in a ArrayList object.
containsAll	(inherited from AbstractCollection)
ensureCapacity	Ensures that the length of a ArrayList object will be no smaller than the specified size.
equals	(inherited from AbstractList)
get	Overridden. Returns the element at the specified index.
hashCode	(inherited from AbstractList)
getClass	(inherited from Object)
indexOf	Overridden. Provides the index of an element in a ArrayList object.
isEmpty	Overridden. Determines whether a ArrayList object is empty.
iterator	(inherited from AbstractList)
lastIndexOf	Overridden. Provides the index of the last occurrence of an element in a ArrayList object.
listIterator	Overloaded. (inherited from AbstractList)
MemberwiseClone	Creates a new instance of a ArrayList object that is a shallow copy of an existing ArrayList object.
remove	Overloaded. Overridden. Removes an element from a ArrayList object.
removeAll	(inherited from AbstractCollection)
removeRange	Overridden. Removes all elements within the specified range from a ArrayList object.
retainAll	(inherited from AbstractCollection)
set	Overridden. Sets an element in a ArrayList object at the specified index.
size	Overridden. Provides the number of elements in a ArrayList object.
subList	(inherited from AbstractList)
toArray	Overloaded. Overridden. Creates a static array containing the elements of a ArrayList object.

toString	(inherited from AbstractCollection)
trimToSize	Trims a ArrayList object to the number of elements it contains.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ArrayList Class

Concepts

java.util Package

ArrayList.add Method

Adds an element to the end of a ArrayList.

Overload List

Name	Description
ArrayList.add (Object)	Adds an element to a ArrayList.
ArrayList.add (int, Object)	Adds an element to the ArrayList object at a specified index.

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.add Method (Object)

Adds an element to a ArrayList.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean add(
    java.lang.Object e);
```

Parameters

e

Return Value

Returns true if the element was successfully added; false otherwise.

Example

This example creates a ArrayList object, and adds the even number objects from 0 to 10 to the list.

```
// Add1.jsl
// ArrayList example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      ArrayList al = new ArrayList();

      for (int i=0; i<=10; i=i+2)
      {
         Integer x = new Integer(i);
         al.add(x);
      }

      System.out.println("The list members are: " + al);
   }
}

/*
Output:
The list members are: [0,2,4,6,8,10]
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.add Method (Int32, Object)

Adds an element to the ArrayList object at a specified index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void add(
    int ix,
    java.lang.Object e);
```

Parameters

ix

The index of the element.

e

The element to be added.

Example

```
// Add2.jsl
// ArrayList example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      ArrayList al = new ArrayList();

      for (int i=0; i<10; i=i+2)
      {
         Integer x = new Integer(i);
         al.add(i/2, x);
         System.out.println("Element " + x
                            + " at index " + i/2);
      }
   }
}

/*
Output:
Element 0 at index 0
Element 2 at index 1
Element 4 at index 2
Element 6 at index 3
Element 8 at index 4
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.addAll Method

Adds all the items in an existing collection into a ArrayList object.

Overload List

Name	Description
ArrayList.addAll (Collection)	Adds all the items in an existing collection into a ArrayList object.
ArrayList.addAll (int, Collection)	Adds all the items in an existing collection to a ArrayList object at the specified index.

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.addAll Method (Collection)

Adds all the items in an existing collection into a ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean addAll(
    java.util.Collection c);
```

Parameters

c

The collection whose items will be added to the ArrayList object.

Return Value

Returns true if the elements were successfully added; false otherwise.

Example

The following example fills an ArrayList object with the contents of another ArrayList object and a HashSet object.

```
// arraylist_addall.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create an ArrayList containing elements 1, 2, 3.
        ArrayList initialArray = new ArrayList();
        initialArray.add(new Integer(1));
        initialArray.add(new Integer(2));
        initialArray.add(new Integer(3));

        // Create a HashSet containing elements 4, 5, 6.
        HashSet initialSet = new HashSet();
        initialSet.add(new Integer(4));
        initialSet.add(new Integer(5));
        initialSet.add(new Integer(6));

        ArrayList newArray = new ArrayList();

        // Add initialArray to the end of newArray.
        boolean addedArray = newArray.addAll(initialArray);
        // Add initialSet to the end of newArray.
        boolean addedSet = newArray.addAll(initialSet);

        // newArray now has elements 1, 2, 3, 4, 5, 6.
        for (int i = 0; i < newArray.size(); i++)
        {
            System.out.println("newArray[" + i + "]: " +
                newArray.get(i));
        }
    }
}

/*
Output:
newArray[0]: 1
newArray[1]: 2
newArray[2]: 3
```

```
  newArray[3]: 4
  newArray[4]: 5
  newArray[5]: 6
  */
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.addAll Method (Int32, Collection)

Adds all the items in an existing collection to a ArrayList object at the specified index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean addAll(
    int ix,
    java.util.Collection c);
```

Parameters

ix

The index where the items will be inserted.

c

The collection whose items will be added to the ArrayList object.

Return Value

Returns true if the items were successfully added; false otherwise.

Example

The following example fills an ArrayList object with the contents of another ArrayList object and a HashSet object.

```
// arraylist_addall_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create an ArrayList containing elements 1, 2, 3.
        ArrayList initialArray = new ArrayList();
        initialArray.add(new Integer(1));
        initialArray.add(new Integer(2));
        initialArray.add(new Integer(3));

        // Create a HashSet containing elements 4, 5, 6.
        HashSet initialSet = new HashSet();
        initialSet.add(new Integer(4));
        initialSet.add(new Integer(5));
        initialSet.add(new Integer(6));

        // Create an ArrayList containing elements 0, 7.
        ArrayList newArray = new ArrayList();
        newArray.add(new Integer(0));
        newArray.add(new Integer(7));

        // Add initialArray to newArray starting at position 1.
        // (After element 0)
        boolean addedArray = newArray.addAll(1, initialArray);
        // Add initialSet to newArray starting at position 4.
        // (After element 3)
        boolean addedSet = newArray.addAll(4, initialSet);

        // newArray now has elements 0, 1, 2, 3, 4, 5, 6, 7.
        for (int i = 0; i < newArray.size(); i++)
        {
            System.out.println("newArray[" + i + "]: " +
                newArray.get(i));
```

```
        }
    }
}

/*
Output:
newArray[0]: 0
newArray[1]: 1
newArray[2]: 2
newArray[3]: 3
newArray[4]: 4
newArray[5]: 5
newArray[6]: 6
newArray[7]: 7
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.clear Method

Removes all items from a ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

Example

The following example populates an ArrayList with a few prime numbers and then clears them all from the ArrayList.

```
// arraylist_clear.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Fill an array with the prime numbers between
        // 1 and 11.
        ArrayList primeNumbers = new ArrayList();
        primeNumbers.add(new Integer(1));
        primeNumbers.add(new Integer(3));
        primeNumbers.add(new Integer(5));
        primeNumbers.add(new Integer(7));
        primeNumbers.add(new Integer(11));

        // Prints out the prime numbers between 1 and 11.
        System.out.println("Before clear...");
        for (int i = 0; i < primeNumbers.size(); i++)
        {
            System.out.println("primeNumbers[" + i + "]: " +
                primeNumbers.get(i));
        }

        // Removes all entries from the ArrayList.
        primeNumbers.clear();

        // Prints no elements.
        System.out.println("After clear...");
        for (int i = 0; i < primeNumbers.size(); i++)
        {
            System.out.println("primeNumbers[" + i + "]: " +
                primeNumbers.get(i));
        }
    }
}

/*
Output:
Before clear...
primeNumbers[0]: 1
primeNumbers[1]: 3
primeNumbers[2]: 5
primeNumbers[3]: 7
primeNumbers[4]: 11
After clear...
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.contains Method

Determines whether an element exists in a ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean contains(
    java.lang.Object e);
```

Parameters

e

The element for which the ArrayList object is to be scanned to determine whether it exists in the collection.

Return Value

Returns true if the element exists in the ArrayList object; false otherwise.

Example

The following example searches an ArrayList to determine whether it contains various prime numbers.

```
// arraylist_contains.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Fill an array with the prime numbers between
        // 1 and 11.
        ArrayList primeNumbers = new ArrayList();
        primeNumbers.add(new Integer(1));
        primeNumbers.add(new Integer(3));
        primeNumbers.add(new Integer(5));
        primeNumbers.add(new Integer(7));
        primeNumbers.add(new Integer(11));

        // Determines whether all the numbers between 1 and 11
        // are prime or not prime.
        for (int i = 1; i <= 11; i++)
        {
            if (primeNumbers.contains(new Integer(i)))
            {
                System.out.println(i + " is a prime number.");
            }
            else
            {
                System.out.println(i + " is NOT a prime number.");
            }
        }
    }
}

/*
Output:
1 is a prime number.
2 is NOT a prime number.
3 is a prime number.
4 is NOT a prime number.
5 is a prime number.
6 is NOT a prime number.
```

```
7 is a prime number.
8 is NOT a prime number.
9 is NOT a prime number.
10 is NOT a prime number.
11 is a prime number.
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.ensureCapacity Method

Ensures that the length of a ArrayList object will be no smaller than the specified size.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void ensureCapacity(
    int minCap);
```

Parameters

minCap

The minimum length for a ArrayList object.

Example

The following example increases the capacity of an ArrayList from 5 to 20 elements.

```
// arraylist_ensurecapacity.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Creates an ArrayList with room for only 5 elements.
        ArrayList smallArray = new ArrayList(5);

        // Increases the capacity of the ArrayList to 20 elements.
        smallArray.ensureCapacity(20);
    }
}
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.get Method

Returns the element at the specified index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object get(
    int ix);
```

Parameters

ix

The index of the element to be returned.

Return Value

The element at the specified index.

Example

The following example creates an ArrayList containing various breeds of dogs. It then calls get passing both valid and invalid indices.

```
// arraylist_get.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Fill an ArrayList with various breeds of dogs.
        ArrayList dogs = new ArrayList();

        Dog fido = new Dog("Golden Retriever");
        Dog fluffy = new Dog("Poodle");
        Dog shadow = new Dog("Basenji");
        Dog miki = new Dog("Yorkshire Terrier");

        dogs.add(fido);
        dogs.add(fluffy);
        dogs.add(shadow);
        dogs.add(miki);

        try
        {
            // Print out the dogs in the ArrayList.
            // oops, goes beyond the end of the ArrayList.
            for (int i = 0; i < 5; i++)
            {
                System.out.println(dogs.get(i).toString());
            }
        }
        catch (IndexOutOfBoundsException ex)
        {
            System.out.println("get was called with an illegal index");
        }
    }
}

public class Dog
{
    public Dog(String breed)
```

```
    {
        this.breed = breed;
    }

    public String toString()
    {
        return breed;
    }

    private String breed;
}

/*
Output:
Golden Retriever
Poodle
Baseni
Yorkshire Terrier
get was called with an illegal index
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.indexOf Method

Provides the index of an element in a ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int indexOf(
    java.lang.Object e);
```

Parameters

e

The element whose index is to be returned.

Return Value

The index of an element in ArrayList, or -1 if the element does not exist.

Example

The following example creates an ArrayList containing various breeds of dogs. It then tries to determine where a certain dog is located within the ArrayList.

```
// arraylist_indexof.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Fill an ArrayList with various breeds of dogs on exhibit
        // at a dog show.
        ArrayList dogs = new ArrayList();

        Dog fido = new Dog("Golden Retriever");
        Dog fluffy = new Dog("Poodle");
        Dog shadow = new Dog("Basenji");
        Dog miki = new Dog("Yorkshire Terrier");

        dogs.add(fido);
        dogs.add(fluffy);
        dogs.add(shadow);
        dogs.add(miki);

        // Where are the barkless dogs?.
        int stageNumber = dogs.indexOf(shadow);
        if (stageNumber > 0)
        {
            System.out.println("Shadow the basenji is on stage " +
                stageNumber);
        }
        else
        {
            System.out.println("There are currently no basenjis " +
                "being shown.");
        }
    }
}

public class Dog
{
    public Dog(String breed)
```

```
    {
        this.breed = breed;
    }

    public String toString()
    {
        return breed;
    }

    private String breed;
}

/*
Output:
Shadow the basenji is on stage 2
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.isEmpty Method

Determines whether a ArrayList object is empty.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isEmpty();
```

Return Value

Returns true if a ArrayList object contains no elements; false otherwise.

Example

The following example shows how to determine if an ArrayList is empty.

```
// arraylist_isempty.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create an empty ArrayList.
        ArrayList arrList = new ArrayList();

        if (arrList.isEmpty())
        {
            System.out.println("The ArrayList is empty");
        }
        else
        {
            // Do something useful with the ArrayList.
        }
    }
}
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.lastIndexOf Method

Provides the index of the last occurrence of an element in a ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int lastIndexOf(
    java.lang.Object e);
```

Parameters

e

The element whose index is to be returned.

Return Value

The index of the last occurrence of an element in a ArrayList object, or -1 if the element does not exist.

Example

The following example creates an ArrayList containing various breeds of dogs. It then tries to determine where a certain dog is located within the ArrayList.

```
// arraylist_lastindexof.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Fill an ArrayList with various breeds of dogs on exhibit
        // at a dog show.
        ArrayList dogs = new ArrayList();

        Dog fido = new Dog("Golden Retriever");
        Dog fluffy = new Dog("Poodle");
        Dog shadow = new Dog("Basenji");
        Dog miki = new Dog("Yorkshire Terrier");

        dogs.add(fido);
        dogs.add(fluffy);
        dogs.add(shadow);
        dogs.add(miki);

        // Where are the barkless dogs?.
        int stageNumber = dogs.lastIndexOf(shadow);
        if (stageNumber > 0)
        {
            System.out.println("Shadow the basenji is on stage " +
                stageNumber);
        }
        else
        {
            System.out.println("There are currently no basenjis " +
                "being shown.");
        }
    }
}

public class Dog
{
    public Dog(String breed)
```

```
    {
        this.breed = breed;
    }

    public String toString()
    {
        return breed;
    }

    private String breed;
}

/*
Output:
Shadow the basenji is on stage 2
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.MemberwiseClone Method

Creates a new instance of a ArrayList object that is a shallow copy of an existing ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

A new instance of ArrayList that is a shallow copy of an existing ArrayList object. The returned object can be safely hard cast to a ArrayList object.

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.remove Method

Removes an element from a ArrayList object.

Overload List

Name	Description
ArrayList.remove (int)	Removes an element at a specified index from ArrayList.
ArrayList.remove (Object)	

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.remove Method (Int32)

Removes an element at a specified index from ArrayList.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object remove(
    int ix);
```

Parameters

ix

The index of the element to be removed.

Return Value

The element at the position next to the removed element.

Example

```
// Add3.jsl
// ArrayList example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      ArrayList al =  new ArrayList();

      for (int i=0; i<10; i=i+2)
      {
         Integer x = new Integer(i);
         al.add(i/2, x);
      }
      System.out.println("The ArrayList members are: " + al);
      al.remove(2);
      System.out.println("The ArrayList members are now: " + al);
   }
}

/*
Output:
The ArrayList members are: [0,2,4,6,8]
The ArrayList members are now: [0,2,6,8]
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.removeRange Method

Removes all elements within the specified range from a ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected void removeRange(
    int fromIx,
    int toIx);
```

Parameters

fromIx

The starting index for the range of elements to be removed.

toIx

The ending index for the range of elements to be removed.

Remarks

The element at toIx is not included in the elements to be removed.

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.set Method

Sets an element in a ArrayList object at the specified index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object set(
    int ix,
    java.lang.Object e);
```

Parameters

ix

The index of the element to be set.

e

The element to be set at the specified index.

Return Value

The element that previously existed at the specified index.

Example

The following example changes an existing element in an ArrayList.

```
// arraylist_set.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create an ArrayList containing a simple sentence.
        ArrayList arrList = new ArrayList();
        arrList.add(new String("Today"));
        arrList.add(new String("is"));
        arrList.add(new String("Thursday"));

        // Change the day to Friday.
        arrList.set(2, new String("Friday"));

        Iterator iter = arrList.iterator();
        while (iter.hasNext())
        {
            System.out.print(iter.next() + " ");
        }
    }
}

/*
Output:
Today is Friday
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.size Method

Provides the number of elements in a ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

Return Value

The number of elements in a ArrayList object.

Example

The following example prints the size of an ArrayList object.

```
// arraylist_size.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create an ArrayList containing elements 1, 2, 3.
        ArrayList initialArray = new ArrayList();
        initialArray.add(new Integer(1));
        initialArray.add(new Integer(2));
        initialArray.add(new Integer(3));

        // Determine the size of the ArrayList.
        int size = initialArray.size();
        System.out.println("ArrayList contains " + size +
            " elements.");
    }
}

/*
Output:
ArrayList contains 3 elements.
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.toArray Method

Creates a static array containing the elements of a ArrayList object.

Overload List

Name	Description
ArrayList.toArray ()	Creates a static array containing the elements of a ArrayList object.
ArrayList.toArray (Object[])	Adds the elements of a ArrayList object into a static array.

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.toArray Method ()

Creates a static array containing the elements of a ArrayList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object[] toArray();
```

Return Value

An array containing the elements of a ArrayList object.

Example

The following example converts an ArrayList into an array and displays the contents of the array.

```
// arraylist_toarray.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create an ArrayList containing elements 1, 2, 3.
        ArrayList initialArray = new ArrayList();
        initialArray.add(new Integer(1));
        initialArray.add(new Integer(2));
        initialArray.add(new Integer(3));

        // Create an array from the ArrayList.
        Object[] arr = initialArray.toArray();

        // Display the contents of the array.
        for (int i = 0; i < arr.length; i++)
        {
            if (arr[i] instanceof Integer)
            {
                System.out.println(arr[i].toString());
            }
        }
    }
}

/*
Output:
1
2
3
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.toArray Method (Object[ ])

Adds the elements of a ArrayList object into a static array.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object[] toArray(
    java.lang.Object[] arr);
```

Parameters

arr

The static array where the elements of a ArrayList object are to be added.

Return Value

A static array containing the elements of a ArrayList object.

Example

The following example replaces the contents of an existing array with the contents of an ArrayList.

```
// arraylist_toarray_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create an ArrayList containing elements 1, 2, 3.
        ArrayList initialArray = new ArrayList();
        initialArray.add(new Integer(1));
        initialArray.add(new Integer(2));
        initialArray.add(new Integer(3));

        // Create an array containing elements 4, 5, 6.
        Object[] arr = { new Integer(4),
                         new Integer(5),
                         new Integer(6) };

        // The existing elements in arr are replaced with the
        // contents of the ArrayList.
        arr = initialArray.toArray(arr);

        // Display the contents of the array.
        for (int i = 0; i < arr.length; i++)
        {
            if (arr[i] instanceof Integer)
            {
                System.out.println(arr[i].toString());
            }
        }
    }
}

/*
Output:
1
2
3
*/
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

ArrayList.trimToSize Method

Trims a ArrayList object to the number of elements it contains.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void trimToSize();
```

Example

The following example creates an ArrayList with a capacity of 100 elements. Three elements are then added to the ArrayList and the ArrayList is trimmed accordingly.

```
// arraylist_trimtosize.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create an ArrayList with capacity 100.
        ArrayList initialArray = new ArrayList(100);

        // Only add three elements to the ArrayList.
        initialArray.add(new Integer(1));
        initialArray.add(new Integer(2));
        initialArray.add(new Integer(3));

        // Trim the ArrayList down to size.
        initialArray.trimToSize();
    }
}
```

See Also

Reference

ArrayList Class

Concepts

ArrayList Members

java.util Package

Arrays Class

A class that contains several methods for sorting, searching, filling and comparing arrays.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.Arrays
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.util.Arrays

See Also

Concepts

Arrays Members

java.util Package

Arrays Members

A class that contains several methods for sorting, searching, filling and comparing arrays.

The following tables list the members exposed by the Arrays type.

Public Constructors

Name	Description
Arrays	Constructs an Arrays object.

Public Methods

Name	Description
asList	Returns a list view of a specified array object.
binarySearch	Overloaded.
equals	Overloaded.
equals	(inherited from Object)
fill	Overloaded.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
sort	Overloaded.
toString	Overridden. Converts an array to a string.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Arrays Class

Concepts

java.util Package

Arrays Constructor

Constructs an Arrays object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Arrays();
```

Example

```
// Construct an arrays object:
Arrays myArray = new Arrays();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays Methods

Public Methods

Name	Description
asList	Returns a list view of a specified array object.
binarySearch	Overloaded.
equals	Overloaded.
equals	(inherited from Object)
fill	Overloaded.
hashCode	(inherited from Object)
getClass	(inherited from Object)
clone	
sort	Overloaded.
toString	Overridden. Converts an array to a string.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Arrays Class

Concepts

java.util Package

Arrays.asList Method

Returns a list view of a specified array object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.List asList(
    java.lang.Object[] arr);
```

Parameters

arr

An array object.

Return Value

The list view of the specified array.

Example

```
// Arr-asList1.jsl
// Arrays.asList example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare an object array:
        Object[] myArr = { "Item1", "Item2", "Item3", "Four", "5" };

        // View the array as a list:
        List lst = Arrays.asList(myArr);

        // Display the list:
        System.out.print(lst);
    }
}

/*
Output:
[Item1, Item2, Item3, Four, 5]
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method

Overload List

Name	Description
Arrays.binarySearch (char[], char)	Searches a char array for a specified character.
Arrays.binarySearch (double[], double)	Searches a double array for a specified value using binary search.
Arrays.binarySearch (short[], short)	Searches an array of short numbers for a specified value using binary search.
Arrays.binarySearch (int[], int)	Searches an int array for a specified value.
Arrays.binarySearch (long[], long)	Searches a long array for a specified value.
Arrays.binarySearch (Object[], Object)	Searches an array of objects for a specified value.
Arrays.binarySearch (byte[], byte)	Searches a byte array for a specified value.
Arrays.binarySearch (float[], float)	Searches a float array for a specified value.
Arrays.binarySearch (Object[], Object, Comparator)	Searches an array of objects for a specified value by using a comparator.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method (Char[ ], Char)

Searches a char array for a specified character.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    char[] arr,
    char e);
```

Parameters

arr

The array to search within.

e

The value to search for.

Return Value

The index of the value, or a negative number if the value was not found.

Example

```
// Arr-BinSrch-ch.jsl
// Arrays.binarySearch example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare and initialize an array:
        char[] myArr = {'1','2','3','4','5'};

        // Search the array:
        int idx = Arrays.binarySearch(myArr,'2');

        // Display the result:
        System.out.print("The element " + myArr[idx] +
                         " is found at index: " + idx);
    }
}

/*
Output:
The element 2 is found at index: 1
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method (Double[ ], Double)

Searches a double array for a specified value using binary search.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    double[] arr,
    double e);
```

Parameters

arr

The array to search within.

e

The value to search for.

Return Value

The index of the value, or a negative number if the value was not found.

Example

```
// Arr-BinSrch-dbl.jsl
// Arrays.binarySearch example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare and initialize an array:
        double[] myArr = {1.5,2.5,3.3,4.3,5.2};

        // Search the array:
        int idx = Arrays.binarySearch(myArr,3.3);

        // Display the result:
        System.out.print("The element " + myArr[idx] +
                         " is found at index: " + idx);
    }
}

/*
Output:
The element 3.3 is found at index: 2
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method (Int16[ ], Int16)

Searches an array of short numbers for a specified value using binary search.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    short[] arr,
    short e);
```

Parameters

arr

The array to search within.

e

The value to search for.

Return Value

The index of the value, or a negative number if the value was not found.

Example

```
// Arr-BinSrch-sh.jsl
// Arrays.binarySearch example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare an array:
        short[] myArr = new short[5];

        // Fill with data:
        for (int i=0; i<myArr.length; i++)
        myArr[i] = (short)(i+1);

        // Search the array:
        int idx = Arrays.binarySearch(myArr,(short)3);

        // Display the result:
        System.out.print("The element " + myArr[idx] +
                         " is found at index: " + idx);
    }
}

/*
Output:
The element 3 is found at index: 2
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method (Int32[ ], Int32)

Searches an int array for a specified value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    int[] arr,
    int e);
```

Parameters

arr

The array to search within.

e

The value to search for.

Return Value

The index of the value, or a negative number if the value was not found.

Example

```
// Arr-BinSrch-int.jsl
// Arrays.binarySearch example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare an array:
        int[] myArr = new int[5];

        // Fill with data:
        for (int i=0; i<myArr.length; i++)
        myArr[i] = i+1;

        // Search the array:
        int idx = Arrays.binarySearch(myArr,3);

        // Display the result:
        System.out.print("The element " + myArr[idx] +
                         " is found at index: " + idx);
    }
}

/*
Output:
The element 3 is found at index: 2
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method (Int64[ ], Int64)

Searches a long array for a specified value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    long[] arr,
    long e);
```

Parameters

arr

The array to search within.

e

The value to search for.

Return Value

The index of the value, or a negative number if the value was not found.

Example

```
// Arr-BinSrch-lng.jsl
// Arrays.binarySearch example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare an array:
        long[] myArr = new long[5];

        // Fill with data:
        for (int i=0; i<myArr.length; i++)
        myArr[i] = i+1;

        // Search the array:
        int idx = Arrays.binarySearch(myArr,3);

        // Display the result:
        System.out.print("The element " + myArr[idx] +
                         " is found at index: " + idx);
    }
}

/*
Output:
The element 3 is found at index: 2
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method (Object[ ], Object)

Searches an array of objects for a specified value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    java.lang.Object[] arr,
    java.lang.Object e);
```

Parameters

arr

The array to search within.

e

The value to search for.

Return Value

The index of the value, or a negative number if the value was not found.

Example

```
// Arr-BinSrch-ob.jsl
// Arrays.binarySearch example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare and initialize an object array:
        Object[] myArr = {"1","2","3","4","5"};

        // Search the array:
        int idx = Arrays.binarySearch(myArr,"3");

        // Display the result:
        System.out.print("The element " + myArr[idx] +
                         " is found at index: " + idx);
    }
}

/*
Output:
The element 3 is found at index: 2
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method (SByte[ ], SByte)

Searches a byte array for a specified value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    byte[] arr,
    byte e);
```

Parameters

arr

The array to search within.

e

The value to search for.

Return Value

The index of the value, or a negative number if the value was not found.

Example

```
// Arr-BinSrch-by.jsl
// Arrays.binarySearch example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare an array:
        byte[] myArr = new byte[5];

        // Fill with data:
        for (int i=0; i<myArr.length; i++)
        myArr[i] = (byte)(i+1);

        // Search the array:
        int idx = Arrays.binarySearch(myArr,(byte)2);

        // Display the result:
        System.out.print("The element " + myArr[idx] +
                         " is found at index: " + idx);
    }
}

/*
Output:
The element 2 is found at index: 1
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method (Single[ ], Single)

Searches a float array for a specified value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    float[] arr,
    float e);
```

Parameters

arr

The array to search within.

e

The value to search for.

Return Value

The index of the value, or a negative number if the value was not found.

Example

```
// Arr-BinSrch-fl.jsl
// Arrays.binarySearch example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare an array:
        float[] myArr = new float[5];

        // Fill with data:
        for (int i=0; i<myArr.length; i++)
        myArr[i] = (float)(i+1);

        // Search the array:
        int idx = Arrays.binarySearch(myArr,3F);

        // Display the result:
        System.out.print("The element " + myArr[idx] +
                         " is found at index: " + idx);
    }
}

/*
Output:
The element 3.0 is found at index: 2
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.binarySearch Method (Object[ ], Object, Comparator)

Searches an array of objects for a specified value by using a comparator.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    java.lang.Object[] arr,
    java.lang.Object e,
    java.util.Comparator comp);
```

Parameters

arr

The array to search within.

e

The value to search for.

comp

The comparator used in ordering the array. The value null is used for natural order.

Return Value

The index of the value, or a negative number if the value was not found.

Example

```
// Arr-BinSrch-obc.jsl
// Arrays.binarySearch example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Declare and initialize an object array:
        Object[] myArr = {"1","2","3","4","5"};

        // Search the array:
        int idx = Arrays.binarySearch(myArr,"3",null);

        // Display the result:
        System.out.print("The element " + myArr[idx] +
                         " is found at index: " + idx);
    }
}

/*
Output:
The element 3 is found at index: 2
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method

Overload List

Name	Description
Arrays.equals (boolean[], boolean[])	Checks if two boolean arrays are equal.
Arrays.equals (char[], char[])	Checks if two char arrays are equal.
Arrays.equals (double[], double[])	Checks if two double arrays are equal.
Arrays.equals (short[], short[])	Checks if two short arrays are equal.
Arrays.equals (int[], int[])	Checks if two int arrays are equal.
Arrays.equals (long[], long[])	Checks if two long arrays are equal.
Arrays.equals (Object[], Object[])	Checks if two Object arrays are equal.
Arrays.equals (byte[], byte[])	Checks if two byte arrays are equal.
Arrays.equals (float[], float[])	Checks if two float arrays are equal.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method (Boolean[ ], Boolean[ ])

Checks if two boolean arrays are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static boolean equals(
    boolean[] a1,
    boolean[] a2);
```

Parameters

a1

The first array.

a2

The second array.

Return Value

true if the two arrays are identical; false otherwise.

Example

```
// Arr-equals-bo.jsl
// Arrays.equals example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare and initialize two arrays:
boolean[] myArr1 = {true,false};
boolean[] myArr2 = {true,false};
// Compare the arrays:
boolean result = Arrays.equals(myArr1,myArr2);
// Display the result:
System.out.print("The result is: " + result);
}
}
/*
Output:
The result is: true
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method (Char[ ], Char[ ])

Checks if two char arrays are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static boolean equals(
    char[] a1,
    char[] a2);
```

Parameters

a1

The first array.

a2

The second array.

Return Value

true if the two arrays are identical; false otherwise.

Example

```
// Arr-equals-ch.jsl
// Arrays.equals example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare and initialize two arrays:
char[] myArr1 = {'A','B'};
char[] myArr2 = {'a','b'};
// Compare the arrays:
boolean result = Arrays.equals(myArr1,myArr2);
// Display the result:
System.out.print("The result is: " + result);
}
}
/*
Output:
The result is: false
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method (Double[ ], Double[ ])

Checks if two double arrays are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static boolean equals(
    double[] a1,
    double[] a2);
```

Parameters

a1

The first array.

a2

The second array.

Return Value

true if the two arrays are identical; false otherwise.

Example

```
// Arr-equals-db.jsl
// Arrays.binarySearch example
import java.util.*;
public class MyClass
{
public static void main()
{
        // Declare and initialize two arrays:
double[] myArr1 = {3.14,3.14};
double[] myArr2 = {3.14,3.14};
// Compare the two arrays:
boolean result = Arrays.equals(myArr1,myArr2);
// Display the result:
System.out.print("The result is: " + result);
}
}
/*
Output:
The result is: true
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method (Int16[ ], Int16[ ])

Checks if two short arrays are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static boolean equals(
    short[] a1,
    short[] a2);
```

Parameters

a1

The first array.

a2

The second array.

Return Value

true if the two arrays are identical; false otherwise.

Example

```
// Arr-equals-sh.jsl
// Arrays.equals example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare and initialize two arrays:
short[] myArr1 = {123,456};
short[] myArr2 = {123,456};
// Compare the arrays:
boolean result = Arrays.equals(myArr1,myArr2);
// Display the result:
System.out.print("The result is: " + result);
}
}
/*
Output:
The result is: true
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method (Int32[ ], Int32[ ])

Checks if two int arrays are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static boolean equals(
    int[] a1,
    int[] a2);
```

Parameters

a1

The first array.

a2

The second array.

Return Value

true if the two arrays are identical; false otherwise.

Example

```
// Arr-equals-in.jsl
// Arrays.equals example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare and initialize two arrays:
int[] myArr1 = {123,456};
int[] myArr2 = {123,456};
// Compare the arrays:
boolean result = Arrays.equals(myArr1,myArr2);
// Display the result:
System.out.print("The result is: " + result);
}
}
/*
Output:
The result is: true
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method (Int64[ ], Int64[ ])

Checks if two long arrays are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static boolean equals(
    long[] a1,
    long[] a2);
```

Parameters

a1

The first array.

a2

The second array.

Return Value

true if the two arrays are identical; false otherwise.

Example

```
// Arr-equals-lng.jsl
// Arrays.equals example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare and initialize two arrays:
long[] myArr1 = {123,456};
long[] myArr2 = {123,456};
// Compare the arrays:
boolean result = Arrays.equals(myArr1,myArr2);
// Display the result:
System.out.print("The result is: " + result);
}
}
/*
Output:
The result is: true
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method (Object[ ], Object[ ])

Checks if two Object arrays are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static boolean equals(
    java.lang.Object[] a1,
    java.lang.Object[] a2);
```

Parameters

a1

The first array.

a2

The second array.

Return Value

true if the two arrays are identical; false otherwise.

Example

```
// Arr-equals-ob.jsl
// Arrays.equals example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare and initialize two arrays:
Object[] myArr1 = {"123","456"};
Object[] myArr2 = {"123","456"};
// Compare the arrays:
boolean result = Arrays.equals(myArr1,myArr2);
// Display the result:
System.out.print("The result is: " + result);
}
}
/*
Output:
The result is: true
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method (SByte[ ], SByte[ ])

Checks if two byte arrays are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static boolean equals(
    byte[] a1,
    byte[] a2);
```

Parameters

a1

The first array.

a2

The second array.

Return Value

true if the two arrays are identical; false otherwise.

Example

```
// Arr-equals-by.jsl
// Arrays.equals example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare and initialize two arrays:
byte[] myArr1 = {2,1};
byte[] myArr2 = {1,2};
// Compare the arrays:
boolean result = Arrays.equals(myArr1,myArr2);
// Display the result:
System.out.print("The result is: " + result);
}
}
/*
Output:
The result is: false
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.equals Method (Single[ ], Single[ ])

Checks if two float arrays are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static boolean equals(
    float[] a1,
    float[] a2);
```

Parameters

a1

The first array.

a2

The second array.

Return Value

true if the two arrays are identical; false otherwise.

Example

```
// Arr-equals-fl.jsl
// Arrays.equals example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare and initialize two arrays:
float[] myArr1 = {123f,456f};
float[] myArr2 = {123,456};
// Compare the arrays:
boolean result = Arrays.equals(myArr1,myArr2);
// Display the result:
System.out.print("The result is: " + result);
}
}
/*
Output:
The result is: true
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method

Overload List

Name	Description
Arrays.fill (boolean[], boolean)	Fills a boolean array with a specific value.
Arrays.fill (char[], char)	Fills a char array with a specific value.
Arrays.fill (double[], double)	Fills a double array with a specific value.
Arrays.fill (short[], short)	Fills a short array with a specific value.
Arrays.fill (int[], int)	Fills an int array with a specific value.
Arrays.fill (long[], long)	Fills a long array with a specific value.
Arrays.fill (Object[], Object)	Fills an array of objects with a specific value.
Arrays.fill (byte[], byte)	Fills a byte array with a specific value.
Arrays.fill (float[], float)	Fills a float array with a specific value.
Arrays.fill (boolean[], int, int, boolean)	Fills a specified range of an array with a boolean value.
Arrays.fill (char[], int, int, char)	Fills a specified range of an array with a char value.
Arrays.fill (double[], int, int, double)	Fills a specified range of an array with a double value.
Arrays.fill (short[], int, int, short)	Fills a specified range of an array with a short value.
Arrays.fill (int[], int, int, int)	Fills a specified range of an array with an int value.
Arrays.fill (long[], int, int, long)	Fills a specified range of an array with a long value.
Arrays.fill (Object[], int, int, Object)	Fills a specified range of an array with an Object value.
Arrays.fill (byte[], int, int, byte)	Fills a specified range of an array with a byte value.
Arrays.fill (float[], int, int, float)	Fills a specified range of an array with a float value.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Boolean[ ], Boolean)

Fills a boolean array with a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    boolean[] arr,
    boolean val);
```

Parameters

arr

The array to fill.

val

The value of each element.

Example

```
// Arr-fill-bo.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
boolean[] myArr = new boolean[4];
// Fill with data:
Arrays.fill(myArr,true);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
true true true true
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Char[ ], Char)

Fills a char array with a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    char[] arr,
    char val);
```

Parameters

arr

The array to fill.

val

The value of each element.

Example

```
// Arr-fill-ch.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
char[] myArr = new char[4];
// Fill with data:
Arrays.fill(myArr,'J');
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
J J J J
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Double[ ], Double)

Fills a double array with a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    double[] arr,
    double val);
```

Parameters

arr

The array to fill.

val

The value of each element.

Example

```
// Arr-fill-db.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
double[] myArr = new double[4];
// Fill with data:
Arrays.fill(myArr,3.14);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
3.14 3.14 3.14 3.14
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Int16[ ], Int16)

Fills a short array with a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    short[] arr,
    short val);
```

Parameters

arr

The array to fill.

val

The value of each element.

Example

```
// Arr-fill-sh.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
short[] myArr = new short[4];
// Fill with data:
Arrays.fill(myArr,(short)123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
123 123 123 123
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Int32[ ], Int32)

Fills an int array with a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    int[] arr,
    int val);
```

Parameters

arr

The array to fill.

val

The value of each element.

Example

```
// Arr-fill-int.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
int[] myArr = new int[4];
// Fill with data:
Arrays.fill(myArr,123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
123 123 123 123
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Int64[ ], Int64)

Fills a long array with a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    long[] arr,
    long val);
```

Parameters

arr

The array to fill.

val

The value of each element.

Example

```
// Arr-fill-lng.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
long[] myArr = new long[4];
// Fill with data:
Arrays.fill(myArr,123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
123 123 123 123
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Object[ ], Object)

Fills an array of objects with a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    java.lang.Object[] arr,
    java.lang.Object val);
```

Parameters

arr

The array to fill.

val

The value of each element.

Example

```
// Arr-fill-ob.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
Object[] myArr = new Object[4];
// Fill with data:
Arrays.fill(myArr,"J#");
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
J# J# J# J#
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (SByte[ ], SByte)

Fills a byte array with a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    byte[] arr,
    byte val);
```

Parameters

arr

The array to fill.

val

The value of each element.

Example

```
// Arr-fill-by.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
byte[] myArr = new byte[4];
// Fill with data:
Arrays.fill(myArr,(byte)123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
123 123 123 123
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Single[ ], Single)

Fills a float array with a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    float[] arr,
    float val);
```

Parameters

arr

The array to fill.

val

The value of each element.

Example

```
// Arr-fill-fl.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
float[] myArr = new float[4];
// Fill with data:
Arrays.fill(myArr,123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
123.0 123.0 123.0 123.0
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Boolean[ ], Int32, Int32, Boolean)

Fills a specified range of an array with a boolean value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    boolean[] arr,
    int fromIx,
    int toIx,
    boolean val);
```

Parameters

arr

The array to fill.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

val

The boolean value.

Example

```
// Arr-fill-boArr.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
boolean[] myArr = new boolean[5];
// Fill with data:
Arrays.fill(myArr,2,4,true);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
false false true true false
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Char[ ], Int32, Int32, Char)

Fills a specified range of an array with a char value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    char[] arr,
    int fromIx,
    int toIx,
    char val);
```

Parameters

arr

The array to fill.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

val

The char value.

Example

```
// Arr-fill-chArr.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
char[] myArr = new char[5];
// Fill with data:
Arrays.fill(myArr,2,4,'J');
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
J J
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Double[ ], Int32, Int32, Double)

Fills a specified range of an array with a double value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    double[] arr,
    int fromIx,
    int toIx,
    double val);
```

Parameters

arr

The array to fill.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

val

The double value.

Example

```
// Arr-fill-dbArr.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
double[] myArr = new double[5];
// Fill with data:
Arrays.fill(myArr,2,4,3.14);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
0.0 0.0 3.14 3.14 0.0
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Int16[ ], Int32, Int32, Int16)

Fills a specified range of an array with a short value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    short[] arr,
    int fromIx,
    int toIx,
    short val);
```

Parameters

arr

The array to fill.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

val

The short value.

Example

```
// Arr-fill-shArr.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
short[] myArr = new short[5];
// Fill with data:
Arrays.fill(myArr,2,4,(short)123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
0 0 123 123 0
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Int32[ ], Int32, Int32, Int32)

Fills a specified range of an array with an int value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    int[] arr,
    int fromIx,
    int toIx,
    int val);
```

Parameters

arr

The array to fill.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

val

The int value.

Example

```
// Arr-fill-intArr.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
int[] myArr = new int[5];
// Fill with data:
Arrays.fill(myArr,2,4,123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
0 0 123 123 0
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Int64[ ], Int32, Int32, Int64)

Fills a specified range of an array with a long value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    long[] arr,
    int fromIx,
    int toIx,
    long val);
```

Parameters

arr

The array to fill.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

val

The long value.

Example

```
// Arr-fill-lngArr.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
long[] myArr = new long[5];
// Fill with data:
Arrays.fill(myArr,2,4,123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
0 0 123 123 0
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Object[ ], Int32, Int32, Object)

Fills a specified range of an array with an Object value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    java.lang.Object[] arr,
    int fromIx,
    int toIx,
    java.lang.Object val);
```

Parameters

arr

The array to fill.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

val

The Object value.

Example

```
// Arr-fill-obArr.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
Object[] myArr = new Object[5];
// Fill with data:
Arrays.fill(myArr,2,4,"J#");
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
null null J# J# null
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (SByte[ ], Int32, Int32, SByte)

Fills a specified range of an array with a byte value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    byte[] arr,
    int fromIx,
    int toIx,
    byte val);
```

Parameters

arr

The array to fill.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

val

The byte value.

Example

```
// Arr-fill-byArr.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
byte[] myArr = new byte[5];
// Fill with data:
Arrays.fill(myArr,2,4,(byte)123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
0 0 123 123 0
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.fill Method (Single[ ], Int32, Int32, Single)

Fills a specified range of an array with a float value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    float[] arr,
    int fromIx,
    int toIx,
    float val);
```

Parameters

arr

The array to fill.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

val

The float value.

Example

```
// Arr-fill-flArr.jsl
// Arrays.fill example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
float[] myArr = new float[5];
// Fill with data:
Arrays.fill(myArr,2,4,123);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
0.0 0.0 123.0 123.0 0.0
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method

Overload List

Name	Description
Arrays.sort (char[])	Sorts the specified char array in an ascending order.
Arrays.sort (double[])	Sorts the specified double array in an ascending order.
Arrays.sort (short[])	Sorts the specified short array in an ascending order.
Arrays.sort (int[])	Sorts the specified int array in an ascending order.
Arrays.sort (long[])	Sorts the specified long array in an ascending order.
Arrays.sort (Object[])	Sorts the specified Object array in an ascending order.
Arrays.sort (byte[])	Sorts the specified byte array in an ascending order.
Arrays.sort (float[])	Sorts the specified float array in an ascending order.
Arrays.sort (Object[], Comparator)	Sorts the specified Object array by using a comparator.
Arrays.sort (char[], int, int)	Sorts a specified range of a char array in an ascending order.
Arrays.sort (double[], int, int)	Sorts a specified range of a double array in an ascending order.
Arrays.sort (short[], int, int)	Sorts a specified range of a short array in an ascending order.
Arrays.sort (int[], int, int)	Sorts a specified range of an int array in an ascending order.
Arrays.sort (long[], int, int)	Sorts a specified range of a long array in an ascending order.
Arrays.sort (Object[], int, int)	Sorts a specified range of an Object array in an ascending order.
Arrays.sort (byte[], int, int)	Sorts a specified range of a byte array in an ascending order.
Arrays.sort (float[], int, int)	Sorts a specified range of a float array in an ascending order.
Arrays.sort (Object[], int, int, Comparator)	Sorts the specified range of an Object array by using a comparator.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Char[ ])

Sorts the specified char array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    char[] arr);
```

Parameters

arr

The array to be sorted.

Example

```
// Arr-chSrt.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
char[] myArr = {'A','C','B','E','D'};
// Sort data:
Arrays.sort(myArr);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
A B C D E
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Double[ ])

Sorts the specified double array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    double[] arr);
```

Parameters

arr

The array to be sorted.

Example

```
// Arr-dbSrt.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
double[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1.0 5.0 6.0 7.0 9.0
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Int16[ ])

Sorts the specified short array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    short[] arr);
```

Parameters

arr

The array to be sorted.

Example

```
// Arr-shSrt.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
short[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1 5 6 7 9
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Int32)

Sorts the specified int array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    int[] arr);
```

Parameters

arr

The array to be sorted.

Example

```
// Arr-intSrt.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
int[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1 5 6 7 9
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Int64)

Sorts the specified long array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    long[] arr);
```

Parameters

arr

The array to be sorted.

Example

```
// Arr-lngSrt.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
long[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1 5 6 7 9
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Object[ ])

Sorts the specified Object array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    java.lang.Object[] arr);
```

Parameters

arr

The array to be sorted.

Example

```
// Arr-ObSrt.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
Object[] myArr = {"A","1.0","Zoo","Z"};
// Sort data:
Arrays.sort(myArr);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1.0 A Z Zoo
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (SByte[ ])

Sorts the specified byte array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    byte[] arr);
```

Parameters

arr

The array to be sorted.

Example

```
// Arr-bySrt.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
byte[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1 5 6 7 9
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Single[ ])

Sorts the specified float array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    float[] arr);
```

Parameters

arr

The array to be sorted.

Example

```
// Arr-flSrt.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
float[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1.0 5.0 6.0 7.0 9.0
*/
```

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Object[ ], Comparator)

Sorts the specified Object array by using a comparator.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    java.lang.Object[] arr,
    java.util.Comparator comp);
```

Parameters

arr

The array to be sorted.

comp

An instance of Comparator.

Example

```
// Arr-ObSrt2.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
Object[] myArr = {"A","1.0","Zoo","Z"};
// Sort data:
Arrays.sort(myArr,null);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1.0 A Z Zoo
*/
```

Remarks

If comp is null, the natural order is used in sorting.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Char[ ], Int32, Int32)

Sorts a specified range of a char array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    char[] arr,
    int fromIx,
    int toIx);
```

Parameters

arr

The array to be sorted.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

Example

```
// Arr-chSrtRange.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
char[] myArr = {'A','C','B','E','D'};
// Sort data:
Arrays.sort(myArr,1,3);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
A B C E D
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Double[ ], Int32, Int32)

Sorts a specified range of a double array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    double[] arr,
    int fromIx,
    int toIx);
```

Parameters

arr

The array to be sorted.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

Example

```
// Arr-dbSrtRange.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
double[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr,1,4);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1.0 5.0 7.0 9.0 6.0
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Int16[ ], Int32, Int32)

Sorts a specified range of a short array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    short[] arr,
    int fromIx,
    int toIx);
```

Parameters

arr

The array to be sorted.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

Example

```
// Arr-shSrtRange.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
short[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr,1,4);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1 5 7 9 6
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Int32[ ], Int32, Int32)

Sorts a specified range of an int array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    int[] arr,
    int fromIx,
    int toIx);
```

Parameters

arr

The array to be sorted.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

Example

```
// Arr-intSrtRange.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
int[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr,1,4);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1 5 7 9 6
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Int64[ ], Int32, Int32)

Sorts a specified range of a long array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    long[] arr,
    int fromIx,
    int toIx);
```

Parameters

arr

The array to be sorted.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

Example

```
// Arr-lngSrtRange.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
long[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr,1,4);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1 5 7 9 6
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Object[ ], Int32, Int32)

Sorts a specified range of an Object array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    java.lang.Object[] arr,
    int fromIx,
    int toIx);
```

Parameters

arr

The array to be sorted.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

Example

```
// Arr-ObjSrtRange.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
Object[] myArr = {"A","1.0","ABC","Zoo","Z"};
// Sort data:
Arrays.sort(myArr,1,4);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
A 1.0 ABC Zoo Z
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (SByte[ ], Int32, Int32)

Sorts a specified range of a byte array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    byte[] arr,
    int fromIx,
    int toIx);
```

Parameters

arr

The array to be sorted.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

Example

```
// Arr-bySrtRange.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
byte[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr,1,4);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1 5 7 9 6
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Single[ ], Int32, Int32)

Sorts a specified range of a float array in an ascending order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    float[] arr,
    int fromIx,
    int toIx);
```

Parameters

arr

The array to be sorted.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

Example

```
// Arr-flSrtRange.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
float[] myArr = {1,9,7,5,6};
// Sort data:
Arrays.sort(myArr,1,4);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
1.0 5.0 7.0 9.0 6.0
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

Arrays.sort Method (Object[ ], Int32, Int32, Comparator)

Sorts the specified range of an Object array by using a comparator.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    java.lang.Object[] arr,
    int fromIx,
    int toIx,
    java.util.Comparator comp);
```

Parameters

arr

The array to be sorted.

fromIx

The starting index (inclusive) of the range.

toIx

The ending index (exclusive) of the range.

comp

An instance of Comparator.

Example

```
// Arr-ObjSrtRangecom.jsl
// Arrays.sort example
import java.util.*;
public class MyClass
{
public static void main()
{
// Declare an array:
Object[] myArr = {"A","1.0","ABC","Zoo","Z"};
// Sort data:
Arrays.sort(myArr,1,4,null);
// Display the array:
for(int i=0; i<myArr.length; i++)
System.out.print(myArr[i] + " ");
}
}
/*
Output:
A 1.0 ABC Zoo Z
*/
```

Remarks

Throws ArrayIndexOutOfBoundsException if the range is invalid.

If comp is null, the natural order is used in sorting.

See Also

Reference

Arrays Class

Concepts

Arrays Members

java.util Package

BitSet Class (J#)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.BitSet
    extends java.lang.Object
    implements java.lang.Cloneable, java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

java.util.BitSet

See Also

Concepts

BitSet Members

java.util Package

BitSet Members (J#)

The following tables list the members exposed by the BitSet type.

Public Constructors

Name	Description
BitSet	Overloaded.

Public Methods

Name	Description
and	
andNot	
clear	
equals	Overridden.
get	
hashCode	Overridden.
getClass	(inherited from Object)
length	
MemberwiseClone	
or	
set	
size	
toString	Overridden.
xor	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BitSet Class

Concepts

java.util Package

BitSet Constructor

Overload List

Name	Description
BitSet ()	
BitSet (int)	

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet Constructor ()

Initializes a new instance of the BitSet Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.BitSet();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet Constructor (Int32)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.BitSet(
    int nbits);
```

Parameters

nbits

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet Methods

Public Methods

Name	Description
and	
andNot	
clear	
equals	Overridden.
get	
hashCode	Overridden.
getClass	(inherited from Object)
length	
clone	
or	
set	
size	
toString	Overridden.
xor	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

BitSet Class

Concepts

java.util Package

BitSet.and Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void and(
    java.util.BitSet bs);
```

Parameters

bs

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.andNot Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void andNot(
    java.util.BitSet bs);
```

Parameters

bs

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.clear Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear(
    int pos);
```

Parameters

pos

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.MemberwiseClone Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.equals Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.get Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean get(
    int pos);
```

Parameters

pos

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.hashCode Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.length Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int length();
```

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.or Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void or(
    java.util.BitSet bs);
```

Parameters

bs

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.set Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void set(
    int pos);
```

Parameters

pos

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.size Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.toString Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

BitSet.xor Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void xor(
    java.util.BitSet bs);
```

Parameters

bs

See Also

Reference

BitSet Class

Concepts

BitSet Members

java.util Package

Calendar Class

Contains methods and members to perform basic operations involving days, weeks, months, and years.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.util.Calendar
    extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable
```

Inheritance Hierarchy

java.lang.Object

java.util.Calendar

java.util.GregorianCalendar

See Also

Concepts

Calendar Members

java.util Package

Calendar Members

Contains methods and members to perform basic operations involving days, weeks, months, and years.

The following tables list the members exposed by the Calendar type.

Public Constructors

Name	Description
Calendar	Overloaded.

Public Fields

Name	Description
AM	A constant indicating that the time is ante meridiem (before noon).
AM_PM	A constant representing the appropriate AM or PM value.
APRIL	A constant representing the month of April.
areFieldsSet	Determines whether all the fields of a Calendar object are set.
AUGUST	A constant representing the month of August.
DATE	A constant representing the date.
DAY_OF_MONTH	A constant representing the day of the month.
DAY_OF_WEEK	A constant representing the day of the week.
DAY_OF_WEEK_IN_MONTH	A constant representing a value for how many times a given day has occurred in the month.
DAY_OF_YEAR	A constant representing the day of the year.
DECEMBER	A constant representing the month of December.
DST_OFFSET	A constant representing the offset for daylight savings time in milliseconds.
ERA	A constant representing the appropriate era (either AD or BC).
FEBRUARY	A constant representing the month of February.
FIELD_COUNT	A constant representing the number of fields that can be set.
fields	An array of all the fields that can be set.
FRIDAY	A constant representing the day Friday.
HOUR	A constant representing the hour.
HOUR_OF_DAY	A constant representing the hour of the day.
isSet	An array of boolean values indicating which fields are set and which fields are not set.

isTimeSet	A value indicating whether the time is set.
JANUARY	A constant representing the month of January.
JULY	A constant representing the month of July.
JUNE	A constant representing the month of June.
MARCH	A constant representing the month of March.
MAY	A constant representing the month of May.
MILLISECOND	A constant representing the milliseconds.
MINUTE	A constant representing the minutes.
MONDAY	A constant representing the day Monday.
MONTH	A constant representing the month.
NOVEMBER	A constant representing the month of November.
OCTOBER	A constant representing the month of October.
PM	A constant indicating that the time is post meridiem (after noon).
SATURDAY	A constant representing the day Saturday.
SECOND	A constant representing the seconds.
SEPTEMBER	A constant representing the month of September.
SUNDAY	A constant representing the day Sunday.
THURSDAY	A constant representing the day Thursday.
time	A field containing the time in milliseconds.
TUESDAY	A constant representing the day Tuesday.
UNDECIMBER	A constant representing the thirteenth month of the year for calendars that have 13 months.
WEDNESDAY	A constant representing the day Wednesday.
WEEK_OF_MONTH	A constant representing the week of the month.
WEEK_OF_YEAR	A constant representing the week of the year.
YEAR	A constant representing the year.
ZONE_OFFSET	A constant representing the offset for the time zone in milliseconds.

Public Methods

Name	Description
add	Adds the specified amount of time to a calendar field.
after	Determines whether the date-time in an instance of a Calendar object is after the date-time in the calendar specified.
before	Determines whether the date-time in an instance of a Calendar object is before the date-time in t he calendar specified.
clear	Overloaded. Clears the fields of an instance of a Calendar object.
complete	Sets all the fields of an instance of a Calendar object.
computeFields	Computes values for all the fields of an instance of a Calendar object.
computeTime	Computes the time for a current instance of a Calendar object.
equals	Overridden. Determines whether two Calendar objects are equal.
get	Gets the value of the specified field.
getAvailableLocales	Gets an array of all the Locale values that are available.
getFirstDayOfWeek	Gets a value representing the first day of the week.
getGreatestMinimum	Gets a value representing the greatest possible minimum value for a given field.
hashCode	(inherited from Object)
getInstance	Overloaded.
getLeastMaximum	Gets a value representing the smallest possible maximum value for a given field.
getMaximum	Gets a value representing the maximum value for a given field.
getMinimalDaysInFirstWeek	Gets a value representing the minimal number of days for the first week of the calendar year.
getMinimum	Gets a value representing the minimum value for a given field.
getTime	Gets a Date representing the time of a Calendar object.
getTimeInMillis	Gets the time of a Calendar object represented in milliseconds.
getTimeZone	Gets the TimeZone of a Calendar object.
getClass	(inherited from Object)
internalGet	Gets the value of a given field.
isLenient	Determines whether a given Calendar is lenient.
isSet	Determines whether a given field is set.

MemberwiseClone	
roll	Modifies the value of a given field by one.
set	Overloaded. Sets various fields of a Calendar object.
setFirstDayOfWeek	Sets a value representing the first day of the week.
setLenient	Sets a value that determines whether a given Calendar is lenient.
setMinimalDaysInFirstWeek	Sets a value representing the minimal number of days for the first week of the calendar year.
setTime	Sets the time of a Calendar object to the given Date.
setTimeInMillis	Sets the time of a Calendar object to a value represented in milliseconds.
setTimeZone	Sets the TimeZone of a Calendar object.
toString	Overridden. Generates a human readable representation of a Calendar object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Calendar Class

Concepts

java.util Package

Calendar Fields

Public Fields

Name	Description
AM	A constant indicating that the time is ante meridiem (before noon).
AM_PM	A constant representing the appropriate AM or PM value.
APRIL	A constant representing the month of April.
areFieldsSet	Determines whether all the fields of a Calendar object are set.
AUGUST	A constant representing the month of August.
DATE	A constant representing the date.
DAY_OF_MONTH	A constant representing the day of the month.
DAY_OF_WEEK	A constant representing the day of the week.
DAY_OF_WEEK_IN_MONTH	A constant representing a value for how many times a given day has occurred in the month.
DAY_OF_YEAR	A constant representing the day of the year.
DECEMBER	A constant representing the month of December.
DST_OFFSET	A constant representing the offset for daylight savings time in milliseconds.
ERA	A constant representing the appropriate era (either AD or BC).
FEBRUARY	A constant representing the month of February.
FIELD_COUNT	A constant representing the number of fields that can be set.
fields	An array of all the fields that can be set.
FRIDAY	A constant representing the day Friday.
HOUR	A constant representing the hour.
HOUR_OF_DAY	A constant representing the hour of the day.
isSet	An array of boolean values indicating which fields are set and which fields are not set.
isTimeSet	A value indicating whether the time is set.
JANUARY	A constant representing the month of January.
JULY	A constant representing the month of July.
JUNE	A constant representing the month of June.

MARCH	A constant representing the month of March.
MAY	A constant representing the month of May.
MILLISECOND	A constant representing the milliseconds.
MINUTE	A constant representing the minutes.
MONDAY	A constant representing the day Monday.
MONTH	A constant representing the month.
NOVEMBER	A constant representing the month of November.
OCTOBER	A constant representing the month of October.
PM	A constant indicating that the time is post meridiem (after noon).
SATURDAY	A constant representing the day Saturday.
SECOND	A constant representing the seconds.
SEPTEMBER	A constant representing the month of September.
SUNDAY	A constant representing the day Sunday.
THURSDAY	A constant representing the day Thursday.
time	A field containing the time in milliseconds.
TUESDAY	A constant representing the day Tuesday.
UNDECIMBER	A constant representing the thirteenth month of the year for calendars that have 13 months.
WEDNESDAY	A constant representing the day Wednesday.
WEEK_OF_MONTH	A constant representing the week of the month.
WEEK_OF_YEAR	A constant representing the week of the year.
YEAR	A constant representing the year.
ZONE_OFFSET	A constant representing the offset for the time zone in milliseconds.

See Also

Reference

Calendar Class

Concepts

java.util Package

Calendar.AM Field

A constant indicating that the time is ante meridiem (before noon).

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int AM;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.AM_PM Field

A constant representing the appropriate AM or PM value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int AM_PM;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.APRIL Field

A constant representing the month of April.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int APRIL;
```

Example

```
// cal-month1.jsl
// Calendar months example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = Calendar.getInstance();

        // Display the months as constants:
        System.out.println(cal.JANUARY);
        System.out.println(cal.FEBRUARY);
        System.out.println(cal.MARCH);
        System.out.println(cal.APRIL);
        System.out.println(cal.MAY);
        System.out.println(cal.JUNE);
        System.out.println(cal.JULY);
        System.out.println(cal.AUGUST);
        System.out.println(cal.SEPTEMBER);
        System.out.println(cal.OCTOBER);
        System.out.println(cal.NOVEMBER);
        System.out.println(cal.DECEMBER);
    }
}

/*
Output:
0
1
2
3
4
5
6
7
8
9
10
11
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.areFieldsSet Field

Determines whether all the fields of a Calendar object are set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected boolean areFieldsSet;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.AUGUST Field

A constant representing the month of August.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int AUGUST;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.DATE Field

A constant representing the date.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int DATE;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.DAY_OF_MONTH Field

A constant representing the day of the month.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int DAY_OF_MONTH;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.DAY_OF_WEEK Field

A constant representing the day of the week.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int DAY_OF_WEEK;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.DAY_OF_WEEK_IN_MONTH Field

A constant representing a value for how many times a given day has occurred in the month.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int DAY_OF_WEEK_IN_MONTH;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.DAY_OF_YEAR Field

A constant representing the day of the year.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int DAY_OF_YEAR;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.DECEMBER Field

A constant representing the month of December.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int DECEMBER;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.DST_OFFSET Field

A constant representing the offset for daylight savings time in milliseconds.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int DST_OFFSET;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.ERA Field

A constant representing the appropriate era (either AD or BC).

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int ERA;
```

Example

```
// cal-era1.jsl
// Calendar.ERA example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Display the era as a constant:
        System.out.println(Calendar.getInstance().ERA); // 0
    }
}

/*
Output:
0
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.FEBRUARY Field

A constant representing the month of February.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int FEBRUARY;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.FIELD_COUNT Field

A constant representing the number of fields that can be set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int FIELD_COUNT;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.fields Field

An array of all the fields that can be set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected int[] fields;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.FRIDAY Field

A constant representing the day Friday.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int FRIDAY;
```

Example

```
// cal-days1.jsl
// Calendar days example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = Calendar.getInstance();

        // Display the months as constants:
        System.out.println("Days of the week:");
        System.out.println(cal.SUNDAY);
        System.out.println(cal.MONDAY);
        System.out.println(cal.TUESDAY);
        System.out.println(cal.WEDNESDAY);
        System.out.println(cal.THURSDAY);
        System.out.println(cal.FRIDAY);
        System.out.println(cal.SATURDAY);
    }
}

/*
Output:
Days of the week:
1
2
3
4
5
6
7
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.HOUR Field

A constant representing the hour.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int HOUR;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.HOUR_OF_DAY Field

A constant representing the hour of the day.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int HOUR_OF_DAY;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.isSet Field

An array of boolean values indicating which fields are set and which fields are not set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected boolean[] isSet;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.isTimeSet Field

A value indicating whether the time is set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected boolean isTimeSet;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.JANUARY Field

A constant representing the month of January.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int JANUARY;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.JULY Field

A constant representing the month of July.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int JULY;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.JUNE Field

A constant representing the month of June.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int JUNE;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.MARCH Field

A constant representing the month of March.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int MARCH;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.MAY Field

A constant representing the month of May.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int MAY;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.MILLISECOND Field

A constant representing the milliseconds.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int MILLISECOND;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.MINUTE Field

A constant representing the minutes.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int MINUTE;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.MONDAY Field

A constant representing the day Monday.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int MONDAY;
```

Example

See the example on FRIDAY.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.MONTH Field

A constant representing the month.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int MONTH;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.NOVEMBER Field

A constant representing the month of November.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int NOVEMBER;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.OCTOBER Field

A constant representing the month of October.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int OCTOBER;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.PM Field

A constant indicating that the time is post meridiem (after noon).

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int PM;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.SATURDAY Field

A constant representing the day Saturday.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int SATURDAY;
```

Example

See the example on FRIDAY.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.SECOND Field

A constant representing the seconds.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int SECOND;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.SEPTEMBER Field

A constant representing the month of September.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int SEPTEMBER;
```

Example

See the example on APRIL.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.SUNDAY Field

A constant representing the day Sunday.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int SUNDAY;
```

Example

See the example on FRIDAY.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.THURSDAY Field

A constant representing the day Thursday.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int THURSDAY;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.time Field

A field containing the time in milliseconds.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected long time;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.TUESDAY Field

A constant representing the day Tuesday.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int TUESDAY;
```

Example

See the example on FRIDAY.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.UNDECIMBER Field

A constant representing the thirteenth month of the year for calendars that have 13 months.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int UNDECIMBER;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.WEDNESDAY Field

A constant representing the day Wednesday.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int WEDNESDAY;
```

Example

See the example on FRIDAY.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.WEEK_OF_MONTH Field

A constant representing the week of the month.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int WEEK_OF_MONTH;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.WEEK_OF_YEAR Field

A constant representing the week of the year.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int WEEK_OF_YEAR;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.YEAR Field

A constant representing the year.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int YEAR;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.ZONE_OFFSET Field

A constant representing the offset for the time zone in milliseconds.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int ZONE_OFFSET;
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar Constructor

Overload List

Name	Description
Calendar ()	Constructs a new instance of a Calendar object.
Calendar (TimeZone, Locale)	Constructs a new instance of a Calendar object using the provided TimeZone and Locale.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar Constructor ()

Constructs a new instance of a Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.Calendar();
```

Example

```
// Create a calendar object:
Calendar cal = Calendar.getInstance();
// or use a subclass:
Calendar cal = new GregorianCalendar();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar Constructor (TimeZone, Locale)

Constructs a new instance of a Calendar object using the provided TimeZone and Locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.Calendar(
    java.util.TimeZone tz,
    java.util.Locale loc);
```

Parameters

tz

The Timezone that the **Calendar** will be initialized with.

loc

The Locale that the **Calendar** will be initialized with.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar Methods

Public Methods

Name	Description
add	Adds the specified amount of time to a calendar field.
after	Determines whether the date-time in an instance of a Calendar object is after the date-time in the calendar specified.
before	Determines whether the date-time in an instance of a Calendar object is before the date-time in t he calendar specified.
clear	Overloaded. Clears the fields of an instance of a Calendar object.
complete	Sets all the fields of an instance of a Calendar object.
computeFields	Computes values for all the fields of an instance of a Calendar object.
computeTime	Computes the time for a current instance of a Calendar object.
equals	Overridden. Determines whether two Calendar objects are equal.
get	Gets the value of the specified field.
getAvailableLocales	Gets an array of all the Locale values that are available.
getFirstDayOfWeek	Gets a value representing the first day of the week.
getGreatestMinimum	Gets a value representing the greatest possible minimum value for a given field.
hashCode	(inherited from Object)
getInstance	Overloaded.
getLeastMaximum	Gets a value representing the smallest possible maximum value for a given field.
getMaximum	Gets a value representing the maximum value for a given field.
getMinimalDaysInFirstWeek	Gets a value representing the minimal number of days for the first week of the calendar year.
getMinimum	Gets a value representing the minimum value for a given field.
getTime	Gets a Date representing the time of a Calendar object.
getTimeInMillis	Gets the time of a Calendar object represented in milliseconds.
getTimeZone	Gets the TimeZone of a Calendar object.
getClass	(inherited from Object)
internalGet	Gets the value of a given field.

isLenient	Determines whether a given Calendar is lenient.
isSet	Determines whether a given field is set.
MemberwiseClone	
roll	Modifies the value of a given field by one.
set	Overloaded. Sets various fields of a Calendar object.
setFirstDayOfWeek	Sets a value representing the first day of the week.
setLenient	Sets a value that determines whether a given Calendar is lenient.
setMinimalDaysInFirstWeek	Sets a value representing the minimal number of days for the first week of the calendar year.
setTime	Sets the time of a Calendar object to the given Date.
setTimeInMillis	Sets the time of a Calendar object to a value represented in milliseconds.
setTimeZone	Sets the TimeZone of a Calendar object.
toString	Overridden. Generates a human readable representation of a Calendar object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Calendar Class

Concepts

java.util Package

Calendar.add Method

Adds the specified amount of time to a calendar field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void add(
    int fld,
    int amount);
```

Parameters

fld

The field to be modified.

amount

The amount to add to the above field.

Example

```
// cal-add1.jsl
// Calendar.add example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a calendar:
        Calendar cal = new GregorianCalendar();

        // Add 4 years to the current calendar:
        cal.add(1,4);
        System.out.println("Year: " + cal.get(Calendar.YEAR));
    }
}

/*
Output:
Year: 2008
*/
```

Remarks

Argument fld can be one of the following defined in class Calendar:

ERA

YEAR

MONTH

WEEK_OF_YEAR

WEEK_OF_MONTH

DATE

DAY_OF_MONTH

DAY_OF_YEAR

DAY_OF_WEEK

DAY_OF_WEEK_IN_MONTH

AM_PM

HOUR

HOUR_OF_DAY

MINUTE

SECOND

MILLISECOND

ZONE_OFFSET

DST_OFFSET

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.after Method

Determines whether the date-time in an instance of a Calendar object is after the date-time in the calendar specified.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean after(
    java.lang.Object cal);
```

Parameters

cal

The calendar to compare to the current object.

Return Value

true if the current calendar contains a later date/time than cal; false otherwise.

Example

```
// cal-after1.jsl
// Calendar.after example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct the current calendar:
        Calendar cal1 = new GregorianCalendar();

        // Construct another calendar:
        Calendar cal2 = new GregorianCalendar(2004,7,10);


        // Check if cal1 is after cal2:
        System.out.println(cal1.after(cal2));
    }
}

/*
Output:
true
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.before Method

Determines whether the date-time in an instance of a Calendar object is before the date-time in the calendar specified.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean before(
    java.lang.Object cal);
```

Parameters

cal

The calendar to compare to the current object.

Return Value

true if the current calendar contains an earlier date-time than cal; false otherwise.

Example

```
// cal-before1.jsl
// Calendar.after example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct the current calendar:
        Calendar cal1 = new GregorianCalendar();

        // Construct another calendar:
        Calendar cal2 = new GregorianCalendar(2004,7,10);

        // Check if cal1 is before cal2:
        System.out.println(cal1.before(cal2));
    }
}

/*
Output:
false
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.clear Method

Clears the fields of an instance of a Calendar object.

Overload List

Name	Description
Calendar.clear ()	Clears all the fields of an instance of a Calendar object.
Calendar.clear (int)	Clears the specified field of an instance of a Calendar object.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.clear Method ()

Clears all the fields of an instance of a Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final void clear();
```

Example

```
// Create a calendar object:
Calendar cal = Calendar.getInstance();
// Clear the year field of the calendar object:
cal.clear(Calendar.YEAR);
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.clear Method (Int32)

Clears the specified field of an instance of a Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final void clear(
    int fld);
```

Parameters

fld

The field to be cleared.

Example

```
// Create a calendar object:
Calendar cal = Calendar.getInstance();
// Clear the field #4:
cal.clear(4);
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.complete Method

Sets all the fields of an instance of a Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected void complete();
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.computeFields Method

Computes values for all the fields of an instance of a Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected abstract void computeFields();
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.computeTime Method

Computes the time for a current instance of a Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected abstract void computeTime();
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.equals Method

Determines whether two Calendar objects are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The Calendar object to compare to.

Return Value

true if the two Calendar objects are equal; false otherwise.

Example

```
// cal-equals1.jsl
// Calendar.Equals example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create two calendar objects:
        Calendar cal2 = new GregorianCalendar(2004,8,10);
        Calendar cal1 = new GregorianCalendar(2004,8,10);

        // Compare the two objects:
        System.out.println(cal1.Equals(cal2));
    }
}

/*
Output:
true
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.get Method

Gets the value of the specified field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final int get(
    int fld);
```

Parameters

fld

The field whose value is to be retrieved.

Return Value

The value of the calendar field specified by fld.

Example

```
// cal-get1.jsl
// Calendar.get example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a default calendar:
        Calendar cal = Calendar.getInstance();

        // Get and display information from the calendar:
        System.out.println("Year: " + cal.get(Calendar.YEAR));
        System.out.println("Month: " + cal.get(Calendar.MONTH));
        System.out.println("Day: " + cal.get(Calendar.DATE));
        System.out.println("Hour: " + cal.get(Calendar.HOUR));
        System.out.println("Minute: " + cal.get(Calendar.MINUTE));
    }
}

/*
Output:
Year: 2004
Month: 8
Day: 14
Hour: 1
Minute: 30
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getAvailableLocales Method

Gets an array of all the Locale values that are available.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.util.Locale[] getAvailableLocales();
```

Return Value

An array of Locale values that are supported by Calendar.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getFirstDayOfWeek Method

Gets a value representing the first day of the week.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getFirstDayOfWeek();
```

Return Value

A value representing the first day of the week.

Example

```
// cal-gefirst.jsl
// Calendar.getFirstDayOfWeek example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a calendar:
        Calendar cal = Calendar.getInstance();

        // Get and display information from the calendar:
        int fdw = cal.getFirstDayOfWeek();

        System.out.print("The first day of the week is: ");

        switch(fdw) {
            case Calendar.SUNDAY:
                System.out.println("Sunday");
                break;
            case Calendar.MONDAY:
                System.out.println("Monday");
                break;
            case Calendar.TUESDAY:
                System.out.println("Tuesday");
                break;
            case Calendar.WEDNESDAY:
                System.out.println("Wednesday");
                break;
            case Calendar.THURSDAY:
                System.out.println("Thursday");
                break;
            case Calendar.FRIDAY:
                System.out.println("Friday");
                break;
            case Calendar.SATURDAY:
                System.out.println("Saturday");
                break;
            default:
                System.out.println("Invalid!");
        }
    }
}

/*
Output:
The first day of the week is: Monday
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getGreatestMinimum Method

Gets a value representing the greatest possible minimum value for a given field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int getGreatestMinimum(
    int fld);
```

Parameters

fld

The field whose greatest possible minimum value is to be retrieved.

Return Value

The greatest possible minimum value for fld.

Example

```
// cal-Min1.jsl
// Calendar.getLeastMinimum example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a GregorianCalendar object:
        Calendar cal = new GregorianCalendar(2004,9,10);

        // Display the greatest min. for field #1:
        System.out.println("The minimum is: " +
                           cal.getGreatestMinimum(1));
    }
}

/*
Output:
The minimum is: 1
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getInstance Method

Overload List

Name	Description
Calendar.getInstance ()	Provides access to a Calendar using the default TimeZone and Locale.
Calendar.getInstance (Locale)	Provides access to a Calendar using the default TimeZone and given Locale.
Calendar.getInstance (TimeZone)	Provides access to a Calendar using the given TimeZone and default Locale.
Calendar.getInstance (TimeZone, Locale)	Provides access to a Calendar using the given TimeZone and given Locale.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getInstance Method ()

Provides access to a Calendar using the default TimeZone and Locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.util.Calendar getInstance();
```

Return Value

A Calendar with the default TimeZone and Locale.

Example

```
// cal-getInst1.jsl
// Calendar getInstance example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object using defaults:
        Calendar cal = Calendar.getInstance();

        // Display the time:
        System.out.println(cal.getTime());

        // Display the time zone offset:
        System.out.println("TZ offset in hours: " +
                  cal.get(Calendar.ZONE_OFFSET)/(1000*60*60));

    }
}

/*
Output:
Tue Sep 14 13:44:29 PDT 2004
TZ offset in hours: -8
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getInstance Method (Locale)

Provides access to a Calendar using the default TimeZone and given Locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.util.Calendar getInstance(
    java.util.Locale aLocale);
```

Parameters

aLocale

The Locale for the Calendar object.

Return Value

A Calendar with the default TimeZone and given Locale.

Example

```
// cal-getInst2.jsl
// Calendar getInstance example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a locale:
        Locale loc = new Locale("en","us");

        // Create a calendar object using loc:
        Calendar cal = Calendar.getInstance(loc);

        // Display the locale and time zone offset:
        System.out.println("Locale: " + loc);
        System.out.println("TZ offset in hours: " +
                  cal.get(Calendar.ZONE_OFFSET)/(1000*60*60));

    }
}

/*
Output:
Locale: en_US
TZ offset in hours: -8
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getInstance Method (TimeZone)

Provides access to a Calendar using the given TimeZone and default Locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.util.Calendar getInstance(
    java.util.TimeZone zone);
```

Parameters

zone

The TimeZone for the Calendar object.

Return Value

A Calendar with the given TimeZone and default Locale.

Example

```
// cal-getInst3.jsl
// Calendar getInstance example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create Pacific time zone with -8 hours offset:
        TimeZone tz = new SimpleTimeZone(-28800000,
                                         "America/Los_Angeles");
        // Create a calendar object using the time zone:
        Calendar cal = Calendar.getInstance(tz);

        // Display the time zone offset in hours:
        System.out.println("TZ offset in hours: " +
                  cal.get(Calendar.ZONE_OFFSET)/(1000*60*60));
    }
}

/*
Output:
TZ offset in hours: -8
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getInstance Method (TimeZone, Locale)

Provides access to a Calendar using the given TimeZone and given Locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.util.Calendar getInstance(
    java.util.TimeZone zone,
    java.util.Locale aLocale);
```

Parameters

zone

The TimeZone for the Calendar object.

aLocale

The Locale for the Calendar object.

Return Value

A Calendar with the given TimeZone and given Locale.

Example

```
// cal-getInst4.jsl
// Calendar.getInstance example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create Pacific time zone with -8 hours offset:
        TimeZone tz = new SimpleTimeZone(-28800000,
                                         "America/Los_Angeles");

        // Create a locale:
        Locale loc = new Locale("en","us");

        // Construct an object using tz and loc:
        Calendar cal = Calendar.getInstance(tz,loc);

        // Display the locale and time zone offset:
        System.out.println("Locale: " + loc);
        System.out.println("TZ offset in hours: " +
                  cal.get(Calendar.ZONE_OFFSET)/(1000*60*60));
    }
}

/*
Output:
Locale: en_US
TZ offset in hours: -8
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getLeastMaximum Method

Gets a value representing the smallest possible maximum value for a given field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int getLeastMaximum(
    int fld);
```

Parameters

fld

The field whose smallest possible maximum value is to be retrieved.

Return Value

The smallest possible maximum value for fld.

Example

```
// cal-Max1.jsl
// Calendar.getLeastMaximum example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = new GregorianCalendar(2004,9,10);

        // Display the least max. for field #1:
        System.out.println("The maximum is: " +
                           cal.getLeastMaximum(1));
    }
}

/*
Output:
The maximum is: 5000000
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getMaximum Method

Gets a value representing the maximum value for a given field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMaximum(
    int fld);
```

Parameters

fld

The field whose maximum value is to be retrieved.

Return Value

The maximum value for fld.

Example

```
// cal-Max2.jsl
// Calendar.getMaximum example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = new GregorianCalendar(2004,9,10);

        // Display the max. for field #1:
        System.out.println("The maximum is: " +
                           cal.getMaximum(1));
    }
}

/*
Output:
The maximum is: 5000000
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getMinimalDaysInFirstWeek Method

Gets a value representing the minimal number of days for the first week of the calendar year.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getMinimalDaysInFirstWeek();
```

Return Value

The minimal number of days for the first week of the calendar year.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getMinimum Method

Gets a value representing the minimum value for a given field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int getMinimum(
    int fld);
```

Parameters

fld

The field whose minimum value is to be retrieved.

Return Value

The minimum value for fld.

Example

```
// cal-Min2.jsl
// Calendar.getMinimum example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = new GregorianCalendar(2004,9,10);

        // Display the least max. for field #1:
        System.out.println("The minimum is: " +
                           cal.getMinimum(1));
    }
}

/*
Output:
The minimum is: 1
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getTime Method

Gets a Date representing the time of a Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final java.util.Date getTime();
```

Return Value

A Date representing the time of a Calendar object.

Example

```
// cal-getTime1.jsl
// Calendar.getTime example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create an object:
        Calendar cal = new GregorianCalendar();

        // Display the time:
        System.out.println(cal.getTime());
    }
}

/*
Output:
Fri Sep 10 14:12:18 PDT 2004
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getTimeInMillis Method

Gets the time of a Calendar object represented in milliseconds.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected long getTimeInMillis();
```

Return Value

The time of a Calendar object represented in milliseconds.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.getTimeZone Method

Gets the TimeZone of a Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.TimeZone getTimeZone();
```

Return Value

The TimeZone of the Calendar object.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.internalGet Method

Gets the value of a given field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected final int internalGet(
    int fld);
```

Parameters

fld

The field whose value is to be returned.

Return Value

The value of fld.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.isLenient Method

Determines whether a given Calendar is lenient.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isLenient();
```

Return Value

true if the Calendar is lenient; false otherwise.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.isSet Method

Determines whether a given field is set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final boolean isSet(
    int fld);
```

Parameters

fld

The field whose status is to be checked.

Return Value

true if fld is set; false otherwise.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.MemberwiseClone Method

Constructs a new instance of a Calendar object that is a shallow copy of the current Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

A clone of the current Calendar object that is a shallow copy.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.roll Method

Modifies the value of a given field by one.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void roll(
    int fld,
    boolean up);
```

Parameters

fld

The field whose value is to be modified.

up

Flag indicating whether the field is to be increased or decreased. true indicates the field is to be increased.

Example

```
// cal-roll1.jsl
// Calendar.roll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = new GregorianCalendar();

        // Display the current year:
        System.out.println("The current year: " + cal.get(Calendar.YEAR));

        // Roll by 1:
        cal.roll(1,true);

        // Display the new year:
        System.out.println("The new year: " + cal.get(Calendar.YEAR));
    }
}

/*
Output:
The current year: 2004
The new year: 2005
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.set Method

Sets various fields of a Calendar object.

Overload List

Name	Description
Calendar.set (int, int)	Sets the value of a calendar field to a new value.
Calendar.set (int, int, int)	Sets an instance of a Calendar object to the given year, month, and day.
Calendar.set (int, int, int, int, int)	Sets an instance of a Calendar object to the given year, month, day, hour, and minute.
Calendar.set (int, int, int, int, int, int)	Sets an instance of a Calendar object to the given year, month, day, hour, minute, and second.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.set Method (Int32, Int32)

Sets the value of a calendar field to a new value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final void set(
    int fld,
    int newValue);
```

Parameters

fld

The field whose value is to be modified.

newValue

The modified value for the field given.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.set Method (Int32, Int32, Int32)

Sets an instance of a Calendar object to the given year, month, and day.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final void set(
    int year,
    int month,
    int date);
```

Parameters

year

The value for the year field.

month

The value for the month field.

date

The value for the date field.

Example

```
// cal-set12.jsl
// Calendar set example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object using defaults:
        Calendar cal = Calendar.getInstance();

        // Set the calendar:
        cal.set(1999,3,5);

        // Display the results:
        System.out.println("Year: " + cal.get(Calendar.YEAR));
        System.out.println("Month: " + cal.get(Calendar.MONTH));
        System.out.println("Day: " + cal.get(Calendar.DATE));
    }
}

/*
Output:
Year: 1999
Month: 3
Day: 5
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.set Method (Int32, Int32, Int32, Int32, Int32)

Sets an instance of a Calendar object to the given year, month, day, hour, and minute.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final void set(
    int year,
    int month,
    int date,
    int hourOfDay,
    int minute);
```

Parameters

year

The value for the year field.

month

The value for the month field.

date

The value for the date field.

hourOfDay

The value for the hour field.

minute

The value for the minute field.

Example

```
// cal-set13.jsl
// Calendar set example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object using defaults:
        Calendar cal = Calendar.getInstance();

        // Set the calendar:
        cal.set(1999,3,5,8,25);

        // Display the results:
        System.out.println("Year: " + cal.get(Calendar.YEAR));
        System.out.println("Month: " + cal.get(Calendar.MONTH));
        System.out.println("Day: " + cal.get(Calendar.DATE));
        System.out.println("Hour: " + cal.get(Calendar.HOUR));
        System.out.println("Minute: " + cal.get(Calendar.MINUTE));
    }
}

/*
Output:
Year: 1999
Month: 3
Day: 5
```

```
Hour: 8
Minute: 25
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.set Method (Int32, Int32, Int32, Int32, Int32, Int32)

Sets an instance of a Calendar object to the given year, month, day, hour, minute, and second.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final void set(
    int year,
    int month,
    int date,
    int hourOfDay,
    int minute,
    int second);
```

Parameters

year

The value for the year field.

month

The value for the month field.

date

The value for the date field.

hourOfDay

The value for the hour field.

minute

The value for the minute field.

second

The value for the second field.

Example

```
// cal-set14.jsl
// Calendar set example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object using defaults:
        Calendar cal = Calendar.getInstance();

        // Set the calendar:
        cal.set(1999,3,5,8,25,15);

        // Display the results:
        System.out.println("Year: " + cal.get(Calendar.YEAR));
        System.out.println("Month: " + cal.get(Calendar.MONTH));
        System.out.println("Day: " + cal.get(Calendar.DATE));
        System.out.println("Hour: " + cal.get(Calendar.HOUR));
        System.out.println("Minute: " + cal.get(Calendar.MINUTE));
        System.out.println("Second: " + cal.get(Calendar.SECOND));
    }
}
```

```
/*
Output:
Year: 1999
Month: 3
Day: 5
Hour: 8
Minute: 25
Second: 15
*/
```

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.setFirstDayOfWeek Method

Sets a value representing the first day of the week.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setFirstDayOfWeek(
    int value);
```

Parameters

value

A value representing the first day of the week.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.setLenient Method

Sets a value that determines whether a given Calendar is lenient.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setLenient(
    boolean lenient);
```

Parameters

lenient

true if the Calendar should be lenient; false otherwise.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.setMinimalDaysInFirstWeek Method

Sets a value representing the minimal number of days for the first week of the calendar year.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setMinimalDaysInFirstWeek(
    int value);
```

Parameters

value

The minimal number of days for the first week of the calendar year.

See Also
Reference
Calendar Class
Concepts
Calendar Members
java.util Package

Calendar.setTime Method

Sets the time of a Calendar object to the given Date.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void setTime(
    java.util.Date date);
```

Parameters

date

The Date to set the time of the Calendar to.

See Also
Reference
Calendar Class
Concepts
Calendar Members
java.util Package

Calendar.setTimeInMillis Method

Sets the time of a Calendar object to a value represented in milliseconds.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected void setTimeInMillis(
    long millis);
```

Parameters

millis

The time of a Calendar object represented in milliseconds.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Calendar.setTimeZone Method

Sets the TimeZone of a Calendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setTimeZone(
    java.util.TimeZone tz);
```

Parameters

tz

The TimeZone for the Calendar object.

See Also

Reference

Calendar Class

Concepts

Calendar Members

java.util Package

Collection Interface

An interface that represents a group of objects or elements. It is implemented by many collection classes such as LinkedList, ArrayList, HashSet and TreeSet.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.Collection
```

See Also

Concepts

Collection Members

java.util Package

Collection Members

An interface that represents a group of objects or elements. It is implemented by many collection classes such as LinkedList, ArrayList, HashSet and TreeSet.

The following tables list the members exposed by the Collection type.

Public Methods

Name	Description
add	Inserts an element into a Collection object.
addAll	Inserts all the elements from one collection into another collection.
clear	Removes all the elements from a Collection object.
contains	Checks if a Collection object contains a specific element.
containsAll	Checks if a Collection object contains all the elements of another collection.
equals	Checks if a Collection object is equal to a specific object.
hashCode	Retrieves the hash code of a Collection.object.
isEmpty	Checks if a Collection object is empty.
iterator	Provides an iterator to iterate over a Collection object.
remove	Removes a specific element from a Collection object.
removeAll	Deletes all the elements from a Collection object that exist is a specified collection.
retainAll	Copies the elements of a specified collection into a Collection object.
size	Retrieves the number of elements in a Collection object.
toArray	Overloaded.

See Also

Reference

Collection Interface

Concepts

java.util Package

Collection Methods

Public Methods

Name	Description
add	Inserts an element into a Collection object.
addAll	Inserts all the elements from one collection into another collection.
clear	Removes all the elements from a Collection object.
contains	Checks if a Collection object contains a specific element.
containsAll	Checks if a Collection object contains all the elements of another collection.
equals	Checks if a Collection object is equal to a specific object.
hashCode	Retrieves the hash code of a Collection.object.
isEmpty	Checks if a Collection object is empty.
iterator	Provides an iterator to iterate over a Collection object.
remove	Removes a specific element from a Collection object.
removeAll	Deletes all the elements from a Collection object that exist is a specified collection.
retainAll	Copies the elements of a specified collection into a Collection object.
size	Retrieves the number of elements in a Collection object.
toArray	Overloaded.

See Also

Reference

Collection Interface

Concepts

java.util Package

Collection.add Method

Inserts an element into a Collection object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean add(
    java.lang.Object e);
```

Parameters

e

The element to be added to Collection.

Return Value

true if the element was inserted successfully; false otherwise.

Example

```
// col-add1.jsl
// Collection.add example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new collection object:
        Collection lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella Abolrous");
        lList.add("Angelina Abolrous");

        // Get and display the elements:
        System.out.println("The elements of the collection are:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(lList.get(i));
    }
}

/*
Output:
The elements of the collection are:
Isabella Abolrous
Angelina Abolrous
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.addAll Method

Inserts all the elements from one collection into another collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean addAll(
    java.util.Collection c);
```

Parameters

c

The name of the collection whose elements to be added.

Return Value

true if the elements were inserted successfully; false otherwise.

Example

```
// col-addAll1.jsl
// Collection.addAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create ArrayList and LinkedList collections:
        Collection aList = new ArrayList();
        Collection lList = new LinkedList();

        // Initialize aList:
        aList.add("Craig");
        aList.add("Sally");

       // Add all the elements of aList to lList:
        lList.addAll(aList);

        // Display lList:
        System.out.println(lList);
    }
}

/*
Output:
[Craig, Sally]
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.clear Method

Removes all the elements from a Collection object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void clear();
```

Example

In this example, you create a linked list object, add two elements to it, and then clear the list. You also print the elements and the size of the list before and after clearing the list.

```
// col-clear.jsl
// Collection.clear example

import java.util.*;

public class MyClass
{
   public static void main(String[] args)
   {
      LinkedList ll = new LinkedList();
      Integer x = new Integer(123);
      Double y = new Double(4.56);

      // Build the list with the first and last elements:
      ll.addFirst(x);
      ll.addLast(y);

      // Print the list:
      System.out.println("The list elements are: " +
                         ll.getFirst() + ", " + ll.getLast());
      // Print the size:
      System.out.println("The size of the list is: "
                         + ll.size());

      // Clear the list:
      ll.clear();
      // Print the new size:
      System.out.println("The size after clearing the list is: "
                          + ll.size());
   }
}

/*
Output:
The list elements are: 123, 4.56
The size of the list is: 2
The size after clearing the list is: 0
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.contains Method

Checks if a Collection object contains a specific element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean contains(
    java.lang.Object e);
```

Parameters

e

The element searched for.

Return Value

true if the element was found, false otherwise.

Example

In this example, you initialize a linked list with two elements, and then you test the list to see if one of the elements exits in it.

```
// col-contains1.jsl
// Collection.contains example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a collection object:
        Collection lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Check if the element "Angelina" exists in the list:
        if(lList.contains("Angelina"))
            System.out.println("The element exists in the list.");
    }
}

/*
Output:
The element exists in the list.
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.containsAll Method

Checks if a Collection object contains all the elements of another collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean containsAll(
    java.util.Collection c);
```

Parameters

c

The name of the other collection.

Return Value

true if all the elements were found; false otherwise.

Example

```
// col-contAll1.jsl
// Collection.containsAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a collection object:
        Collection lList =  new LinkedList();

        // Add some elements:
        lList.add("Sam");
        lList.add("Michelle");
        lList.add("Jim");
        lList.add("Virginia");

        // Create a new list lList1:
        LinkedList lList1 =  new LinkedList();

        // Add all the elements of lList to lList1:
        lList1.addAll(lList);

        // Confirm that the elements of IList exist in IList1:
        if(lList1.containsAll(lList))
            System.out.println(
                  "All the elements exist in the new list.");
    }
}

/*
Output:
All the elements exist in the new list.
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.equals Method

Checks if a Collection object is equal to a specific object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean equals(
    java.lang.Object c);
```

Parameters

c

The object to compare to.

Return Value

true if the two elements are identical; false otherwise.

Example

```
// col-eq1.jsl
// Collection.Equals example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a collection object lList:
        Collection lList =  new LinkedList();

        // Add some elements:
        lList.add("Sam");
        lList.add("Michelle");
        lList.add("Jim");
        lList.add("Virginia");

        // Create a new collection object lList1:
        LinkedList lList1 =  new LinkedList();

        // Add all the element of lList to lList1:
        lList1.addAll(lList);

        // Check if the two lists are identical:
        if(lList1.Equals(lList))
            System.out.println("The two lists are the same.");
    }
}

/*
Output:
The two lists are the same.
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.hashCode Method

Retrieves the hash code of a Collection.object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int hashCode();
```

Return Value

An integer representing the hash code of the Collection object.

Example

```
// col-getHash1.jsl
// Collection.GetHashCode example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a collection object:
        Collection lList =  new LinkedList();

        // Add some elements:
        lList.add(new Double(5.25));
        lList.add(new Byte((byte)127));
        lList.add(null);
        lList.add(new Integer(55));

        // Print the hash code:
        System.out.println("The hash code is: " + lList.GetHashCode());
    }
}

/*
Output:
The hash code is: -1898639464
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.isEmpty Method

Checks if a Collection object is empty.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isEmpty();
```

Return Value

true if the Collection object is empty; false otherwise.

Example

```
// col-isEmp1.jsl
// Collection.isEmpty example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Sam");
        lList.add("Michelle");
        lList.add("Jim");
        lList.add("Virginia");

        // Clear the list:
        lList.clear();

        // Check if the list is empty:
        if(lList.isEmpty())
            System.out.println("The list is now empty.");
    }
}

/*
Output:
The list is now empty.
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.iterator Method

Provides an iterator to iterate over a Collection object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Iterator iterator();
```

Return Value

An iterator that can be used over the Collection object.

Example

```
// col-iterator1.jsl
// Collection.iterator example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      Collection hSet = new HashSet();

      // Add some elements to the HashSet:
      hSet.add("1 ");
      hSet.add("2 ");
      hSet.add("3 ");
      hSet.add("4 ");

      // Retrieve an iterator to the hashset:
      Iterator iter = hSet.iterator();

      // Extract elements from the list using the iterator.
      // Note that the elements may not follow the order in which they
      // are added to HashSet.
      while(iter.hasNext())
      {
         System.out.print(iter.next());
         iter.remove();
      }
   }
}
/*
Output:
3 1 4 2
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.remove Method

Removes a specific element from a Collection object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean remove(
    java.lang.Object e);
```

Parameters

e

The element to be removed.

Return Value

true if the element was removed successfully; false otherwise.

Example

```
// col-rem1.jsl
// Collection.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Anabella");

        // Display the elements:
        System.out.print("The old list: "+ lList);

        // Remove the element #3:
        lList.remove("Anabella");
        System.out.println(
               "\nThe element \"Anabella\" has been removed.");

        // Display the new elements:
        System.out.print("The new list: " + lList);
    }
}

/*
Output:
The old list: [Isabella, Angelina, Pille, Anabella]
The element "Anabella" has been removed.
The new list: [Isabella, Angelina, Pille]
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

Collection.removeAll Method

Deletes all the elements from a Collection object that exist is a specified collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean removeAll(
    java.util.Collection c);
```

Parameters

c

The collection of elements to be deleted.

Return Value

true if the elements were successfully deleted; false otherwise.

Example

In this example, you create and initialize two collections, aList and lList. Then you remove all the elements from the lList that exist in the aList. The output displays the new contents of lList.

```
// col-remA1.jsl
// Collection.removeAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create two collections:
        LinkedList lList =  new LinkedList();
        ArrayList aList =  new ArrayList();

        // Add some elements to lList:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Hazem");

        // Add some elements to aList:
        aList.add("Isabella");
        aList.add("Angelina");

        // Remove aList members:
        lList.removeAll(aList);

        // Display lList:
        System.out.println("The collection is: " + lList);
    }
}

/*
Output:
The collection is: [Pille, Hazem]
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.retainAll Method

Copies the elements of a specified collection into a Collection object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean retainAll(
    java.util.Collection c);
```

Parameters

c

The collection whose elements are copied.

Return Value

true if the list has changed after the call; false otherwise.

Example

In this example, you create and initialize two collections, aList and lList. Then you copy the elements of aList into lList and display both collections.

```
// col-retA1.jsl
// Collection.retainAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create two collections:
        LinkedList lList =  new LinkedList();
        ArrayList aList =  new ArrayList();

        // Add some elements to lList:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Hazem");

        // Add some elements to aList:
        aList.add("Isabella");
        aList.add("Angelina");

        // Retain elements from aList to lList:
        lList.retainAll(aList);

        // Display the collection:
        System.out.println("The aList collection is: " + aList);
        System.out.println("The lList collection is: " + lList);

    }
}

/*
Output:
The aList collection is: [Isabella, Angelina]
The lList collection is: [Isabella, Angelina]
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.size Method

Retrieves the number of elements in a Collection object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int size();
```

Return Value

The number of elements in the Collection object.

Example

```
// col-size1.jsl
// Collection.size example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Costruct a new Collection object:
        Collection lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Camelia");

        // Display the size of the list:
        System.out.println("The size of the collection is: " +
                           lList.size());
    }
}

/*
Output:
The size of the collection is: 4
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.toArray Method

Overload List

Name	Description
Collection.toArray ()	Converts a Collection to an array.
Collection.toArray (Object[])	Copies the elements of a Collection into specified array object.

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.toArray Method ()

Converts a Collection to an array.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object[] toArray();
```

Return Value

The array object that contains the collection elements.

Example

```
// Col-toArr2.jsl
// Collection.toArray example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a new Collection object:
        Collection lList =  new LinkedList();

         // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Camelia");

        // Create an array from the list:
        Object[] s = lList.toArray();

        // Display the array elements:
        System.out.println("The array elements are:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + s[i]);
    }
}

/*
Output:
The array elements are:
0 = Isabella
1 = Angelina
2 = Pille
3 = Camelia
*/
```

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collection.toArray Method (Object[ ])

Copies the elements of a Collection into specified array object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object[] toArray(
    java.lang.Object[] arr);
```

Parameters

arr

The array object that receives the elements of the Collection object.

Return Value

The array object that contains the Collection elements.

Example

```
// col-toArr2.jsl
// Collection.toArray example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a new Collection object:
        Collection lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Camelia");

        // Create an array "s" from the list:
        Object[] s = lList.toArray();

        // Copy "s" to another object array "s1":
        Object[] s1 = lList.toArray(s);

        // Display the array objects:
        System.out.println("The array objects are:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + s1[i]);
    }
}

/*
Output:
The array objects are:
0 = Isabella
1 = Angelina
2 = Pille
3 = Camelia
*/
```

Remarks

Throws NullPointerException if the specified array is null.

See Also

Reference

Collection Interface

Concepts

Collection Members

java.util Package

Collections Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.Collections
    extends java.lang.Object
```

Inheritance Hierarchy

java.lang.Object

java.util.Collections

See Also

Concepts

Collections Members

java.util Package

Collections Members

The following tables list the members exposed by the Collections type.

Public Constructors

Name	Description
Collections	

Public Fields

Name	Description
EMPTY_LIST	
EMPTY_SET	

Public Methods

Name	Description
binarySearch	Overloaded.
copy	
enumeration	
equals	(inherited from Object)
fill	
hashCode	(inherited from Object)
getClass	(inherited from Object)
max	Overloaded.
min	Overloaded.
nCopies	
reverse	
reverseOrder	
shuffle	Overloaded.
singleton	
sort	Overloaded.
synchronizedCollection	
synchronizedList	

synchronizedMap	
synchronizedSet	
synchronizedSortedMap	
synchronizedSortedSet	
toString	Overridden.
unmodifiableCollection	
unmodifiableList	
unmodifiableMap	
unmodifiableSet	
unmodifiableSortedMap	
unmodifiableSortedSet	

Protected Methods

Name	Description
finalize	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

Collections Class

Concepts

java.util Package

Collections Fields

Public Fields

Name	Description
EMPTY_LIST	
EMPTY_SET	

See Also

Reference

Collections Class

Concepts

java.util Package

Collections.EMPTY_LIST Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.List EMPTY_LIST;
```

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.EMPTY_SET Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Set EMPTY_SET;
```

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections Constructor

Initializes a new instance of the Collections Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Collections();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections Methods

Public Methods

Name	Description
binarySearch	Overloaded.
copy	
enumeration	
equals	(inherited from Object)
fill	
hashCode	(inherited from Object)
getClass	(inherited from Object)
max	Overloaded.
min	Overloaded.
nCopies	
reverse	
reverseOrder	
shuffle	Overloaded.
singleton	
sort	Overloaded.
synchronizedCollection	
synchronizedList	
synchronizedMap	
synchronizedSet	
synchronizedSortedMap	
synchronizedSortedSet	
toString	Overridden.
unmodifiableCollection	
unmodifiableList	

unmodifiableMap	
unmodifiableSet	
unmodifiableSortedMap	
unmodifiableSortedSet	

Protected Methods

Name	Description
finalize	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

Collections Class

Concepts

java.util Package

Collections.binarySearch Method

Overload List

Name	Description
Collections.binarySearch (List, Object)	
Collections.binarySearch (List, Object, Comparator)	

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.binarySearch Method (List, Object)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    java.util.List ls,
    java.lang.Object e);
```

Parameters

ls

e

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.binarySearch Method (List, Object, Comparator)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static int binarySearch(
    java.util.List ls,
    java.lang.Object e,
    java.util.Comparator comp);
```

Parameters

ls

e

comp

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.copy Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void copy(
    java.util.List dest,
    java.util.List src);
```

Parameters

dest

src

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.enumeration Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Enumeration enumeration(
    java.util.Collection c);
```

Parameters

c

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.fill Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void fill(
    java.util.List ls,
    java.lang.Object e);
```

Parameters

ls

e

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.max Method

Overload List

Name	Description
Collections.max (Collection)	
Collections.max (Collection, Comparator)	

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.max Method (Collection)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Object max(
    java.util.Collection c);
```

Parameters

c

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.max Method (Collection, Comparator)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Object max(
    java.util.Collection c,
    java.util.Comparator comp);
```

Parameters

c

comp

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.min Method

Overload List

Name	Description
Collections.min (Collection)	
Collections.min (Collection, Comparator)	

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.min Method (Collection)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Object min(
    java.util.Collection c);
```

Parameters

c

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.min Method (Collection, Comparator)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.lang.Object min(
    java.util.Collection c,
    java.util.Comparator comp);
```

Parameters

c

comp

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.nCopies Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.List nCopies(
    int n,
    java.lang.Object e);
```

Parameters

n

e

See Also
Reference
Collections Class
Concepts
Collections Members
java.util Package

Collections.reverse Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void reverse(
    java.util.List ls);
```

Parameters
ls

See Also
Reference
Collections Class
Concepts
Collections Members
java.util Package

Collections.reverseOrder Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Comparator reverseOrder();
```

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.shuffle Method

Overload List

Name	Description
Collections.shuffle (List)	
Collections.shuffle (List, Random)	

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.shuffle Method (List)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void shuffle(
    java.util.List ls);
```

Parameters

ls

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.shuffle Method (List, Random)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void shuffle(
    java.util.List ls,
    java.util.Random r);
```

Parameters

ls

r

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.singleton Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Set singleton(
    java.lang.Object e);
```

Parameters

e

See Also
Reference
Collections Class
Concepts
Collections Members
java.util Package

Collections.sort Method

Overload List

Name	Description
Collections.sort (List)	
Collections.sort (List, Comparator)	

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.sort Method (List)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    java.util.List ls);
```

Parameters

ls

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.sort Method (List, Comparator)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static void sort(
    java.util.List ls,
    java.util.Comparator comp);
```

Parameters

ls

comp

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.synchronizedCollection Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Collection synchronizedCollection(
    java.util.Collection c);
```

Parameters

c

See Also
Reference
Collections Class
Concepts
Collections Members
java.util Package

Collections.synchronizedList Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.List synchronizedList(
    java.util.List ls);
```

Parameters

ls

See Also
Reference
Collections Class
Concepts
Collections Members
java.util Package

Collections.synchronizedMap Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Map synchronizedMap(
    java.util.Map m);
```

Parameters

m

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.synchronizedSet Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Set synchronizedSet(
    java.util.Set s);
```

Parameters

s

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.synchronizedSortedMap Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.SortedMap synchronizedSortedMap(
    java.util.SortedMap m);
```

Parameters

m

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.synchronizedSortedSet Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.SortedSet synchronizedSortedSet(
    java.util.SortedSet s);
```

Parameters

s

See Also
Reference
Collections Class
Concepts
Collections Members
java.util Package

Collections.unmodifiableCollection Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Collection unmodifiableCollection(
    java.util.Collection c);
```

Parameters

c

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.unmodifiableList Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.List unmodifiableList(
    java.util.List ls);
```

Parameters

ls

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.unmodifiableMap Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Map unmodifiableMap(
    java.util.Map m);
```

Parameters

m

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.unmodifiableSet Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Set unmodifiableSet(
    java.util.Set s);
```

Parameters

s

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.unmodifiableSortedMap Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.SortedMap unmodifiableSortedMap(
    java.util.SortedMap m);
```

Parameters

m

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Collections.unmodifiableSortedSet Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.SortedSet unmodifiableSortedSet(
    java.util.SortedSet s);
```

Parameters

s

See Also

Reference

Collections Class

Concepts

Collections Members

java.util Package

Comparator Interface

Contains methods to compare objects.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.Comparator
```

See Also

Concepts

Comparator Members

java.util Package

Comparator Members

Contains methods to compare objects. It is implemented by the abstract class Collator.

The following tables list the members exposed by the Comparator type.

Public Methods

Name	Description
compare	Compares two string objects for equality.
equals	Checks if an object is equal to the current object.

See Also

Reference

Comparator Interface

Concepts

java.util Package

Comparator Methods

Public Methods

Name	Description
compare	Compares two string objects for equality.
equals	Checks if an object is equal to the current object.

See Also

Reference

Comparator Interface

Concepts

java.util Package

Comparator.compare Method

Compares two objects for relative order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int compare(
    java.lang.Object o1,
    java.lang.Object o2);
```

Parameters

o1

The first object to be compared.

o2

The second object to be compared.

Return Value

Negative value if o1 is less than o2.

Zero if o1 and o2 are equal.

Positive value if o1 is greater than o2.

Example

In this example, you compare the three strings, two strings at a time. You get three different results for three comparisons.

```
// comp-comp1.jsl
// Comparator.compare example

import java.text.*;
import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Construct a Collator object(implements Comparator):
        Collator col = Collator.getInstance();

        // Declare some string objects:
        String s1 = "C#";
        String s2 = "J#";
        String s3 = "C#";

        // Compare strings and display the results:
        System.out.println("The comparison result for s1 & s2 = " +
                           col.compare(s1,s2));
        System.out.println("The comparison result for s2 & s1 = " +
                           col.compare(s2,s1));
        System.out.println("The comparison result for s1 & s3 = " +
                           col.compare(s1,s3));
    }
}

/*
Output:
The comparison result for s1 & s2 = -1
The comparison result for s2 & s1 = 1
The comparison result for s1 & s3 = 0
```

```
*/
```

Remarks

Throws java.lang.ClassCastException if the types of the two objects are different or cannot be compared with this method.

See Also

Reference

Comparator Interface

Concepts

Comparator Members

java.util Package

Comparator.equals Method

Checks if an object is equal to the current object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to the current object.

Return Value

true if the two objects are identical; false otherwise.

Example

In this example you compare two collator objects; one of them is a clone of the other. The result of the comparison is true.

```
// comp-eq1.jsl
// Comparator.Equals example

import java.text.*;

public class Program
{
    public static void main(String[] args)
    {
        // Construct a Collator object(implements Comparator):
        Collator o1 = Collator.getInstance();

        // Clone the Collator object:
        Object o2 = o1.clone();

        // Compare the two objects and display the result:
        System.out.println("The comparison result is: " +
                           o1.equals(o2));
    }
}

/*
Output:
The comparison result is: true
*/
```

See Also

Reference

Comparator Interface

Concepts

Comparator Members

java.util Package

ConcurrentModificationException Class

The exception that is thrown when two threads attempt to modify a collection concurrently.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.ConcurrentModificationException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.util.ConcurrentModificationException

See Also

Concepts

ConcurrentModificationException Members

java.util Package

ConcurrentModificationException Members

The exception that is thrown when two threads attempt to modify a collection concurrently.

The following tables list the members exposed by the ConcurrentModificationException type.

Public Constructors

Name	Description
ConcurrentModificationException	Overloaded. Initializes a new instance of a ConcurrentModificationException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ConcurrentModificationException Class

Concepts

java.util Package

ConcurrentModificationException Constructor

Initializes a new instance of a ConcurrentModificationException object.

Overload List

Name	Description
ConcurrentModificationException ()	Initializes a new instance of a ConcurrentModificationException object.
ConcurrentModificationException (String)	Initializes a new instance of a ConcurrentModificationException with the given message.
ConcurrentModificationException (SerializationInfo, StreamingContext)	Initializes a new instance of a ConcurrentModificationException object during deserialization.
ConcurrentModificationException (String, Exception)	Initializes a new instance of a ConcurrentModificationException with the given message and inner exception.

See Also

Reference

ConcurrentModificationException Class

Concepts

ConcurrentModificationException Members

java.util Package

ConcurrentModificationException Constructor ()

Initializes a new instance of a ConcurrentModificationException object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ConcurrentModificationException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ConcurrentModificationException Class

Concepts

ConcurrentModificationException Members

java.util Package

ConcurrentModificationException Constructor (String)

Initializes a new instance of a ConcurrentModificationException with the given message.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ConcurrentModificationException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also
Reference
ConcurrentModificationException Class
Concepts
ConcurrentModificationException Members
java.util Package

ConcurrentModificationException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a ConcurrentModificationException object during deserialization.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.ConcurrentModificationException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

ConcurrentModificationException Class

Concepts

ConcurrentModificationException Members

java.util Package

ConcurrentModificationException Constructor (String, Exception)

Initializes a new instance of a ConcurrentModificationException with the given message and inner exception.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ConcurrentModificationException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the ConcurrentModificationException being thrown.

See Also

Reference

ConcurrentModificationException Class

Concepts

ConcurrentModificationException Members

java.util Package

ConcurrentModificationException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ConcurrentModificationException Class

Concepts

java.util Package

ConcurrentModificationException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

ConcurrentModificationException Class

Concepts

java.util Package

Date Class (J#)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public class java.sql.Date
    extends java.util.Date
```

Inheritance Hierarchy

java.lang.Object

  java.util.Date

   java.sql.Date

See Also

Concepts

Date Members

java.sql Package

Date Members (J#)

The following tables list the members exposed by the Date type.

Public Constructors

Name	Description
Date	Overloaded.

Public Methods

Name	Description
after	Determines whether the Date object is after a given date. (inherited from Date)
before	Determines whether the Date object is before a given date. (inherited from Date)
compareTo	Compares two dates. (inherited from Date)
equals	Determines whether two Date objects are equal. (inherited from Date)
getDate	Gets a value indicating the date. (inherited from Date)
getDay	Gets a value indicating the day of the week. (inherited from Date)
hashCode	Generates a hash of the date. (inherited from Date)
getHours	Overridden.
getMinutes	Overridden.
getMonth	Gets a value indicating the month. (inherited from Date)
getSeconds	Overridden.
getTime	Gets a value indicating the time in milliseconds since epoch. (inherited from Date)
getTimezoneOffset	Gets a value indicating the time zone offset in milliseconds. (inherited from Date)
getClass	(inherited from Object)
getYear	Gets a value indicating the year. (inherited from Date)
clone	(inherited from Date)
setDate	Sets a value indicating the date. (inherited from Date)
setHours	Overridden.
setMinutes	Overridden.
setMonth	Sets a value indicating the month. (inherited from Date)

setSeconds	Overridden.
setTime	Sets the time to the value represented in milliseconds since epoch. (inherited from Date)
setYear	Sets a value indicating the year. (inherited from Date)
ToBoolean	Converts an instance of a Date object to a Boolean. (inherited from Date)
ToByte	Converts an instance of a Date object to a Byte. (inherited from Date)
ToChar	Converts an instance of a Date object to a Char. (inherited from Date)
ToDateTime	Converts an instance of a Date object to a DateTime. (inherited from Date)
ToDecimal	Converts an instance of a Date object to a Decimal. (inherited from Date)
ToDouble	Converts an instance of a Date object to a Double. (inherited from Date)
toGMTString	Displays a value representing the date/time of a Date object in GMT. (inherited from Date)
ToInt16	Converts an instance of a Date object to an Int16. (inherited from Date)
ToInt32	Converts an instance of a Date object to an Int32. (inherited from Date)
ToInt64	Converts an instance of a Date object to an Int64. (inherited from Date)
toLocaleString	Displays a value representing the date/time of a Date object displayed in the format of current locale. (inherited from Date)
ToSByte	Converts an instance of a Date object to a SByte. (inherited from Date)
ToSingle	Converts an instance of a Date object to a Single. (inherited from Date)
toString	Overloaded. Overridden.
ToType	Converts an instance of a Date object to a given Type. (inherited from Date)
ToUInt16	Converts an instance of a Date object to an UInt16. (inherited from Date)
ToUInt32	Converts an instance of a Date object to an UInt32. (inherited from Date)
ToUInt64	Converts an instance of a Date object to an UInt64. (inherited from Date)
valueOf	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Date Class

Concepts

java.sql Package

Date Constructor (J#)

Overload List

Name	Description
Date ()	Constructs a new instance of a Date object.
Date (long)	Constructs a new instance of a Date object and initializes it using the milliseconds since epoch.
Date (String)	Constructs a new instance of a Date object and initializes it using the given string.
Date (int, int, int)	Constructs a new instance of a Date object and initializes it with the given year, month, and day.
Date (int, int, int, int, int)	Constructs a new instance of a Date object and initializes it with the given year, month, day, hours, and minutes.
Date (int, int, int, int, int, int)	Constructs a new instance of a Date object and initializes it with the given year, month, day, hours, minutes, and seconds

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date Constructor () (J#)

Constructs a new instance of a Date object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Date();
```

Example

```
// da-ctor1.jsl
// Date.ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Display the date:
        System.out.println(d);
    }
}

/*
Output:
Wed Sep 08 13:19:55 PDT 2004
*/
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date Constructor (Int64) (J#)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.sql.Date(
    long msSinceEpoch);
```

Parameters

msSinceEpoch

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date Constructor (String) (J#)

Constructs a new instance of a Date object and initializes it using the given string.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Date(
    java.lang.String dateStr);
```

Parameters

dateStr

A string representation of the date to be set. The format of this string is dependent on the current Locale.

Example

```
// da-ctor2.jsl
// Date.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        Date d = new Date("9/8/2004");
        System.out.println(d);
    }
}

/*
Output:
Wed Sep 08 00:00:00 PDT 2004
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date Constructor (Int32, Int32, Int32) (J#)

Constructs a new instance of a Date object and initializes it with the given year, month, and day.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Date(
    int year,
    int month,
    int day);
```

Parameters

year

A value indicating the year. 1900 is always added to the value passed to this parameter.

month

A value indicating the month, where 0 represents January and 11 represents December.

day

A value indicating the day. The value should be between 1 and 31, or else it will wrap.

Example

```
// da-ctor4.jsl
// Date.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date: Dec. 1st, 2004
        Date d = new Date(104,11,1);  //   2004-1900=104

        System.out.println(d);
    }
}

/*
Output:
Wed Dec 01 00:00:00 PST 2004
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date Constructor (Int32, Int32, Int32, Int32, Int32) (J#)

Constructs a new instance of a Date object and initializes it with the given year, month, day, hours, and minutes.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Date(
    int year,
    int month,
    int day,
    int hours,
    int minutes);
```

Parameters

year

A value indicating the year. 1900 is always added to the value passed to this parameter.

month

A value indicating the month, where 0 represents January and 11 represents December.

day

A value indicating the day. The value should be between 1 and 31, or else it will wrap.

hours

A value indicating the hours. The value should be between 0 and 23, or else it will wrap.

minutes

A value indicating the minutes. The value should be between 0 and 59, or else it will wrap.

Example

```
// da-ctor5.jsl
// Date.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date: Jun. 3rd, 8:25 AM, 2002.
        Date d = new Date(102,5,3,8,25);

        System.out.println(d);
    }
}

/*
Output:
Mon Jun 03 08:25:00 PDT 2002
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date Constructor (Int32, Int32, Int32, Int32, Int32, Int32) (J#)

Constructs a new instance of a Date object and initializes it with the given year, month, day, hours, minutes, and seconds

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Date(
    int year,
    int month,
    int day,
    int hours,
    int minutes,
    int seconds);
```

Parameters

year

A value indicating the year. 1900 is always added to the value passed to this parameter.

month

A value indicating the month, where 0 represents January and 11 represents December.

day

A value indicating the day. The value should be between 1 and 31, or else it will wrap.

hours

A value indicating the hours. The value should be between 0 and 23, or else it will wrap.

minutes

A value indicating the minutes. The value should be between 0 and 59, or else it will wrap.

seconds

A value indicating the seconds. The value should be between 0 and 59, or else it will wrap.

Example

```
// da-ctor6.jsl
// Date.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date: Jun. 3rd, 8:25:24 AM, 2002.
        Date d = new Date(102,5,3,8,25,24);

        System.out.println(d);
    }
}

/*
Output:
Mon Jun 03 08:25:24 PDT 2002
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date Methods (J#)

Public Methods

Name	Description
after	Determines whether the Date object is after a given date. (inherited from Date)
before	Determines whether the Date object is before a given date. (inherited from Date)
compareTo	Compares two dates. (inherited from Date)
equals	Determines whether two Date objects are equal. (inherited from Date)
getDate	Gets a value indicating the date. (inherited from Date)
getDay	Gets a value indicating the day of the week. (inherited from Date)
hashCode	Generates a hash of the date. (inherited from Date)
getHours	Overridden.
getMinutes	Overridden.
getMonth	Gets a value indicating the month. (inherited from Date)
getSeconds	Overridden.
getTime	Gets a value indicating the time in milliseconds since epoch. (inherited from Date)
getTimezoneOffset	Gets a value indicating the time zone offset in milliseconds. (inherited from Date)
getClass	(inherited from Object)
getYear	Gets a value indicating the year. (inherited from Date)
clone	(inherited from Date)
setDate	Sets a value indicating the date. (inherited from Date)
setHours	Overridden.
setMinutes	Overridden.
setMonth	Sets a value indicating the month. (inherited from Date)
setSeconds	Overridden.
setTime	Sets the time to the value represented in milliseconds since epoch. (inherited from Date)
setYear	Sets a value indicating the year. (inherited from Date)

ToBoolean	Converts an instance of a Date object to a Boolean. (inherited from Date)
ToByte	Converts an instance of a Date object to a Byte. (inherited from Date)
ToChar	Converts an instance of a Date object to a Char. (inherited from Date)
ToDateTime	Converts an instance of a Date object to a DateTime. (inherited from Date)
ToDecimal	Converts an instance of a Date object to a Decimal. (inherited from Date)
ToDouble	Converts an instance of a Date object to a Double. (inherited from Date)
toGMTString	Displays a value representing the date/time of a Date object in GMT. (inherited from Date)
ToInt16	Converts an instance of a Date object to an Int16. (inherited from Date)
ToInt32	Converts an instance of a Date object to an Int32. (inherited from Date)
ToInt64	Converts an instance of a Date object to an Int64. (inherited from Date)
toLocaleString	Displays a value representing the date/time of a Date object displayed in the format of current locale. (inherited from Date)
ToSByte	Converts an instance of a Date object to a SByte. (inherited from Date)
ToSingle	Converts an instance of a Date object to a Single. (inherited from Date)
toString	Overloaded. Overridden.
ToType	Converts an instance of a Date object to a given Type. (inherited from Date)
ToUInt16	Converts an instance of a Date object to an UInt16. (inherited from Date)
ToUInt32	Converts an instance of a Date object to an UInt32. (inherited from Date)
ToUInt64	Converts an instance of a Date object to an UInt64. (inherited from Date)
valueOf	

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Date Class

Concepts

java.sql Package

Date.after Method

Determines whether the Date object is after a given date.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean after(
    java.util.Date when);
```

Parameters

when

The Date object to be compared against.

Return Value

true if an instance of a Date object is after when; false otherwise.

Example

```
// da-after1.jsl
// Date.after example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct two dates:
        Date d1 = new Date(102,5,3);
        Date d2 = new Date(99,4,2);

        // Compare the dates:
        boolean b = d1.after(d2);

        System.out.println(b);
    }
}

/*
Output:
true
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.before Method

Determines whether the Date object is before a given date.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean before(
    java.util.Date when);
```

Parameters

when

The Date object to be compared against.

Return Value

true if an instance of a Date object is before when; false otherwise.

Example

```
// da-before1.jsl
// Date.after example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct two dates:
        Date d1 = new Date(102,5,3);
        Date d2 = new Date(99,4,2);

        // Compare the dates:
        boolean b = d1.before(d2);

        System.out.println(b);
    }
}

/*
Output:
false
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.clone Method

Creates an instance of a Date object that is a deep copy.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

A deep copy of an instance of a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.compareTo Method

Compares two dates.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int compareTo(
    java.lang.Object obj);
```

Parameters

obj

A Date object to compare against.

Return Value

-1 if an instance of a Date object is before obj

0 if the two dates are equal

1 if an instance of a Date object is after obj.

Example

```
// da-compar1.jsl
// Date.compareTo example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct two dates:
        Date d1 = new Date(102,5,3);
        Date d2 = new Date(99,4,2);

        // Compare the dates:
        System.out.println(d1.compareTo(d2));   // 1
        System.out.println(d1.compareTo(d1));   // 0
        System.out.println(d2.compareTo(d1));   // -1
    }
}

/*
Output:
1
0
-1
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.equals Method

Determines whether two Date objects are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

A Date object to compare against.

Return Value

true if the two Date objects are equal; false otherwise.

Example

```
// da-Equals1.jsl
// Date.Equals example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct two dates:
        Date d1 = new Date(102,5,3);
        Date d2 = new Date(99,4,2);

        // Compare the dates:
        System.out.println(d1.Equals(d2));
    }
}

/*
Output:
false
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.getDate Method

Gets a value indicating the date.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getDate();
```

Return Value

A value indicating the date. The returned value will be between 1 and 31.

Example

```
// da-getDate1.jsl
// Date.getDate example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Get today's number:
        System.out.println("Today is: " + d.getDate());
    }
}

/*
Output:
8
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.getDay Method

Gets a value indicating the day of the week.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getDay();
```

Return Value

A value indicating the day of the week. The returned value will be between 0 and 6.

Example

```
// da-getDay1.jsl
// Date.getDay example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date("11-Dec-1314");

        // Get the day number:
        System.out.println("Date part in 11-Dec-1314 is: " +
            d.getDay());
    }
}

/*
Output:
Date part in 11-Dec-1314 is: 11
*/
```

Remarks

The days are: Sat: 0, Mon: 1, Tue: 2, Wed: 3, Thu: 4, Fri: 5, Sat: 6.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.hashCode Method

Generates a hash of the date.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

A hash value of the date.

Example

```
// da-HashCode1.jsl
// Date.GetHashCode example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Get the hash code:
        System.out.println("The hash code is: " + d.GetHashCode());
    }
}

/*
Output:
The hash code is: -538450330
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.getHours Method (J#)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getHours();
```

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.getMinutes Method (J#)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getMinutes();
```

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.getMonth Method

Gets a value indicating the month.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getMonth();
```

Return Value

A value indicating the month. The returned value will be between 0 and 11.

Example

```
// da-getMon1.jsl
// Date.getMonth example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Get the month:
        System.out.println("The month is: " + d.getMonth());
    }
}

/*
Output:
The month is: 8
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.getSeconds Method (J#)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public int getSeconds();
```

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.getTime Method

Gets a value indicating the time in milliseconds since epoch.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public long getTime();
```

Return Value

A value indicating the time in milliseconds since epoch.

Example

```
// da-gettim1.jsl
// Date.getTime example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Get the time in ms:
        System.out.println("The time is: " + d.getTime());
    }
}

/*
Output:
The time is: 1094678953321
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.getTimezoneOffset Method

Gets a value indicating the time zone offset in milliseconds.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getTimezoneOffset();
```

Return Value

A value indicating the time zone offset in milliseconds.

Example

```
// da-gettimz1.jsl
// Date.getTimezoneOffset example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Get the timezone offset:
        System.out.println("The time offset is: " +
                           d.getTimezoneOffset());
    }
}

/*
Output:
The time offset is: 420
*/
```

Remarks

The time offset returned is relative to UTC.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.getYear Method

Gets a value indicating the year.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getYear();
```

Return Value

A value indicating the year. The returned value will be the current year - 1900.

Example

```
// da-getY1.jsl
// Date.getYear example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Get the year (The year - 1900):
        System.out.println("The year is: " + d.getYear());
    }
}

/*
Output:
The year is: 104
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.parse Method

Parses a date written in string format.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static long parse(
    java.lang.String s);
```

Parameters

s

A representation of a Date in string format. The format for this string will vary according to the value of the current Locale.

Return Value

A value indicating the time of the given string in milliseconds since epoch.

Example

```
// da-parse1.jsl
// Date.parse example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Parse the date:
        System.out.println("The time in ms is: " +
                           d.parse("9/8/2004"));
    }
}

/*
Output:
The time in ms is: 1094626800000
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.setDate Method

Sets a value indicating the date.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setDate(
    int day);
```

Parameters

day

A value indicating the date. The value should be between 1 and 31, or else it will wrap.

Example

```
// da-setDate1.jsl
// Date.setDate example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Set the day:
        d.setDate(8);
        System.out.println("The day is set.");
    }
}

/*
Output:
The day is set.
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.setHours Method (J#)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setHours(
    int hour);
```

Parameters

hour

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.setMinutes Method (J#)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setMinutes(
    int minute);
```

Parameters

minute

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.setMonth Method

Sets a value indicating the month.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setMonth(
    int month);
```

Parameters

month

A value indicating the month. The value should be between 0 and 11, or else it will wrap.

Example

```
// da-setMon1.jsl
// Date.setMonth example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Set the month:
        d.setMonth(9);
        System.out.println("The month is set.");
    }
}

/*
Output:
The month is set.
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.setSeconds Method (J#)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public void setSeconds(
    int second);
```

Parameters

second

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.setTime Method

Sets the time to the value represented in milliseconds since epoch.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setTime(
    long time);
```

Parameters

time

The time, in milliseconds, since epoch.

Example

```
// da-settim1.jsl
// Date.setTime example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Set the time:
        d.setTime(10946268);
        System.out.println("The time is set.");
    }
}

/*
Output:
The time is set.
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.setYear Method

Sets a value indicating the year.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setYear(
    int year);
```

Parameters

year

A value indicating the year. This value should be the current year - 1900.

Example

```
// da-setYear1.jsl
// Date.setYear example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Set the year:
        d.setYear(1900 + 104);
        System.out.println("The year is set to: " + d.getYear());
    }
}

/*
Output:
The year is set to: 2004
*/
```

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToBoolean Method

Converts an instance of a Date object to a Boolean.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean ToBoolean(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Boolean object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToByte Method

Converts an instance of a Date object to a Byte.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public System.Byte ToByte(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Byte object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToChar Method

Converts an instance of a Date object to a Char.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public char ToChar(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Char object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToDateTime Method

Converts an instance of a Date object to a DateTime.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public System.DateTime ToDateTime(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A DateTime object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToDecimal Method

Converts an instance of a Date object to a Decimal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public System.Decimal ToDecimal(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Decimal object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToDouble Method

Converts an instance of a Date object to a Double.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public double ToDouble(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Double object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.toGMTString Method

Displays a value representing the date/time of a Date object in GMT.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toGMTString();
```

Return Value

A value representing the date/time of a Date object in GMT, or Greenwich Mean Time.

Example

```
// da-toGMT1.jsl
// Date.toGMTString example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // To GMT time:
        System.out.println("The GMT time is: " + d.toGMTString());
    }
}

/*
Output:
The GMT time is: 8 Sep 2004 22:53:04 GMT
*/
```

Remarks

GMT is sometimes referred to as UTC, or Universal Time Coordinated.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToInt16 Method

Converts an instance of a Date object to an Int16.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public short ToInt16(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An Int16 object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToInt32 Method

Converts an instance of a Date object to an Int32.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int ToInt32(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An Int32 object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToInt64 Method

Converts an instance of a Date object to an Int64.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public long ToInt64(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An Int64 object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.toLocaleString Method

Displays a value representing the date/time of a Date object displayed in the format of current locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toLocaleString();
```

Return Value

A value representing the date/time of a Date object displayed in the format of the current locale.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToSByte Method

Converts an instance of a Date object to a SByte.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public byte ToSByte(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An SByte object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToSingle Method

Converts an instance of a Date object to a Single.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public float ToSingle(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A Single object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.toString Method (J#)

Overload List

Name	Description
Date.ToString ()	Displays a human readable representation of a Date object.
Date.ToString (IFormatProvider)	Converts an instance of a Date object to a String.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.toString Method () (J#)

Package: java.sql

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

Date Class

Concepts

Date Members

java.sql Package

Date.toString Method (IFormatProvider) (J#)

Converts an instance of a Date object to a String.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String ToString(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

A String object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToType Method

Converts an instance of a Date object to a given Type.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object ToType(
    System.Type conversionType,
    System.IFormatProvider provider);
```

Parameters

conversionType

The Type to convert the Date object into.

provider

Provides information needed for formatting.

Return Value

An object of type conversionType representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToUInt16 Method

Converts an instance of a Date object to an UInt16.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public System.UInt16 ToUInt16(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An UInt16 object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToUInt32 Method

Converts an instance of a Date object to an UInt32.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public System.UInt32 ToUInt32(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An UInt32 object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.ToUInt64 Method

Converts an instance of a Date object to an UInt64.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public System.UInt64 ToUInt64(
    System.IFormatProvider provider);
```

Parameters

provider

Provides information needed for formatting.

Return Value

An UInt64 object representing a Date object.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Date.UTC Method

Calculates the value, in milliseconds, of the provided date/time converted to UTC.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static long UTC(
    int year,
    int month,
    int day,
    int hours,
    int minutes,
    int seconds);
```

Parameters

year

A value indicating the year. This value should be the current year - 1900.

month

A value indicating the month, where 0 represents January and 11 represents December.

day

A value indicating the day. The value should be between 1 and 31, or else it will wrap.

hours

A value indicating the hours. The value should be between 0 and 23, or else it will wrap.

minutes

A value indicating the minutes. The value should be between 0 and 59, or else it will wrap.

seconds

A value indicating the seconds. The value should be between 0 and 59, or else it will wrap.

Return Value

The time, represented in milliseconds, of the provided date/time, converted to UTC.

Example

```
// da-DateUTC.jsl
// Date.Date.UTC example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a date object:
        Date d = new Date();   // Current date

        // Calculate UTC:
        long l = d.UTC(
                       1900+140,   // year
                       8,          // month
                       3,          // day
                       16,         // hours
                       2,          // minutes
                       30          // seconds
                       );
```

```
        // Display the date:
        System.out.println("The date in ms is: " + l);
    }
}

/*
Output:
The date in ms is: 62188444950000
*/
```

Remarks

UTC, or Universal Time Coordinated, is also sometimes referred to as GMT, or Greenwich Mean Time.

See Also

Reference

Date Class

Concepts

Date Members

java.util Package

Dictionary Class

An abstract class that contains a map of elements associated with keys. Both elements and keys are non-null objects.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.util.Dictionary
    extends java.lang.Object
```

Remarks

This class is extended by the Hashtable class.

Inheritance Hierarchy

java.lang.Object

java.util.Dictionary

java.util.Hashtable

See Also

Concepts

Dictionary Members

java.util Package

Dictionary Members

An abstract class that contains a map of elements associated with keys. Both elements and keys are non-null objects.

The following tables list the members exposed by the Dictionary type.

Public Constructors

Name	Description
Dictionary	This constructor is used by subclasses.

Public Methods

Name	Description
elements	Returns an enumeration that contains the values in a Dictionary object.
equals	(inherited from Object)
get	Retrieves an element associated with a specified key in a Dictionary object.
hashCode	(inherited from Object)
getClass	(inherited from Object)
isEmpty	Checks if a Dictionary object is empty.
keys	Returns an enumeration of the keys in a Dictionary object.
put	Adds an element at a specific key entry to a Dictionary object.
remove	Deletes an element associated with a specific key from a Dictionary object.
size	Retrieves the number of the key-value mappings of a Dictionary object.
toString	Overridden. Generates the string representation of a Hashtable object.

Protected Methods

Name	Description
finalize	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

Dictionary Class

Concepts

java.util Package

Dictionary Constructor

This constructor is used by subclasses.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Dictionary();
```

Example

```
// Create a new hashtable object:
Dictionary ht = new Hashtable();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Dictionary Class

Concepts

Dictionary Members

java.util Package

Dictionary Methods

Public Methods

Name	Description
elements	Returns an enumeration that contains the values in a Dictionary object.
equals	(inherited from Object)
get	Retrieves an element associated with a specified key in a Dictionary object.
hashCode	(inherited from Object)
getClass	(inherited from Object)
isEmpty	Checks if a Dictionary object is empty.
keys	Returns an enumeration of the keys in a Dictionary object.
put	Adds an element at a specific key entry to a Dictionary object.
remove	Deletes an element associated with a specific key from a Dictionary object.
size	Retrieves the number of the key-value mappings of a Dictionary object.
toString	Overridden. Generates the string representation of a Hashtable object.

Protected Methods

Name	Description
finalize	(inherited from Object)
MemberwiseClone	Performs a shallow copy of the members.

See Also

Reference

Dictionary Class

Concepts

java.util Package

Dictionary.elements Method

Returns an enumeration that contains the values in a Dictionary object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Enumeration elements();
```

Return Value

The current value at the key entry.

Example

```
// Dic-put1.jsl
// Dictionary.put example
import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        System.out.println("The current value at \"4\" is " + ht.put("4", "Eve Kennedy"));

        // Add a key that is already present.
        System.out.println("The current value at \"4\" is " + ht.put("4", "Jack"));
    }
}

/*
Output:
The current value at “4” is null
The current value at “4” is Eve Kennedy
*/
```

See Also

Reference

Dictionary Class

Concepts

Dictionary Members

java.util Package

Dictionary.get Method

Retrieves an element associated with a specified key in a Dictionary object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object get(
    java.lang.Object key);
```

Parameters

key

The key of the element.

Return Value

The element associated with key.

Example

```
// Dic-get1.jsl
// Dictionary.get example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Dictionary ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Camelia Solomon");
        ht.put("6","Nick Fredette");

       // Get the elemnt associated with the key "6":
        System.out.println(ht.get("6"));
    }
}

/*
Output:
Nick Fredette
*/
```

See Also

Reference

Dictionary Class

Concepts

Dictionary Members

java.util Package

Dictionary.isEmpty Method

Checks if a Dictionary object has no key/value pairs.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isEmpty();
```

Return Value

true if the Dictionary object has no key/value pairs; false otherwise.

Example

```
// Dic-isempt1.jsl
// Dictionary.isEmpty example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Camelia Solomon");

        // Clear the elements:
        ht.clear();

        // Check if the object is empty:
        System.out.println(ht.isEmpty());
    }
}

/*
Output:
true
*/
```

See Also

Reference

Dictionary Class

Concepts

Dictionary Members

java.util Package

Dictionary.keys Method

Returns an enumeration of the keys in a Dictionary object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Enumeration keys();
```

Return Value

An enumeration of the keys in the Dictionary object.

Example

```
// Dic-keys1.jsl
// Dictionary.keys example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Dictionary ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        // Create an enumeration of the table keys:
        Enumeration htEnum = ht.keys();

        // Enumerate over the key values:
        System.out.println("The table keys are:");
        while (htEnum.hasMoreElements())
        {
            System.out.println(htEnum.nextElement());
        }
    }
}

/*
Output:
The table keys are:
4
2
1
*/
```

See Also

Reference

Dictionary Class

Concepts

Dictionary Members

java.util Package

Dictionary.put Method

Adds an element at a specific key entry to a Dictionary object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object put(
    java.lang.Object key,
    java.lang.Object element);
```

Parameters

key

The key entry at which the element is added.

element

The element to be added.

Return Value

The element to be added to Dictionary and the key entry.

Example

```
// Dic-put1.jsl
// Dictionary.put example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        System.out.println("The current value at \"4\" is " + ht.put("4", "Eve Kennedy"));
        // Add a key that is already present:
        System.out.println("The current value at \"4\" is " + ht.put("4", "Jack"));
    }
}

/*
Output:
The current value at “4” is null
The current value at “4” is Eve Kennedy
*/
```

See Also

Reference

Dictionary Class

Concepts

Dictionary Members

java.util Package

Dictionary.remove Method

Removes the element with the specified key from the Dictionary object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object remove(
    java.lang.Object key);
```

Parameters

key

The key of the element to remove.

Return Value

The value associated with the specified key.

Example

```
// Dic-rem1.jsl
// Dictionary.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Dictionary ht = new Hashtable();

        // Add some elements:
        ht.put("3","Sally Abolrous");
        ht.put("2","Craig Combel");
        ht.put("5","Pille Mandla");

        // Remove an object:
        System.out.println("The following object has been removed:\n" +
                           ht.remove("5"));
    }
}

/*
Output:
The following object has been removed:
Pille Mandla
*/
```

See Also

Reference

Dictionary Class

Concepts

Dictionary Members

java.util Package

Dictionary.size Method

Retrieves the number of the key-value mappings of a Dictionary object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int size();
```

Return Value

The number of the key-value mappings of the Dictionary object.

Example

```
// Dic-size1.jsl
// Dictionary.size example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Dictionary ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        // Display the table size:
        System.out.println("The size of the table is: " + ht.size());
    }
}

/*
Output:
The size of the table is: 3
*/
```

See Also

Reference

Dictionary Class

Concepts

Dictionary Members

java.util Package

EmptyStackException Class

The exception that is thrown when attempting to pop an element off of an empty stack.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.EmptyStackException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.util.EmptyStackException

See Also

Concepts

EmptyStackException Members

java.util Package

EmptyStackException Members

The exception that is thrown when attempting to pop an element off of an empty stack.

The following tables list the members exposed by the EmptyStackException type.

Public Constructors

Name	Description
EmptyStackException	Overloaded. Initializes a new instance of an EmptyStackException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

EmptyStackException Class

Concepts

java.util Package

EmptyStackException Constructor

Initializes a new instance of an EmptyStackException object.

Overload List

Name	Description
EmptyStackException ()	Initializes a new instance of an EmptyStackException object.
EmptyStackException (SerializationInfo, StreamingContext)	Initializes a new instance of an EmptyStackException object during deserialization.
EmptyStackException (String, Exception)	Initializes a new instance of an EmptyStackException object with the given message and inner exception.

See Also

Reference

EmptyStackException Class

Concepts

EmptyStackException Members

java.util Package

EmptyStackException Constructor ()

Initializes a new instance of an EmptyStackException object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.EmptyStackException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

EmptyStackException Class

Concepts

EmptyStackException Members

java.util Package

EmptyStackException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of an EmptyStackException object during deserialization.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.EmptyStackException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also
Reference
EmptyStackException Class
Concepts
EmptyStackException Members
java.util Package

EmptyStackException Constructor (String, Exception)

Initializes a new instance of an EmptyStackException object with the given message and inner exception.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.EmptyStackException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the EmptyStackException being thrown.

See Also

Reference

EmptyStackException Class

Concepts

EmptyStackException Members

java.util Package

EmptyStackException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

EmptyStackException Class

Concepts

java.util Package

EmptyStackException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

EmptyStackException Class

Concepts

java.util Package

Enumeration Interface

Provides an efficient mechanism to traverse the elements of various collections.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.Enumeration
```

Example

The following example demonstrates how to use an Enumeration to enumerate over the elements in a Hashtable.

```
// en-Enum1.jsl
// Enumeration example

import java.util.*;

class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashTable:
        Hashtable ht =  new Hashtable();

        // Add some elements:
        ht.put("1","Eve Kennedy");
        ht.put("2","Audrey Esteban");
        ht.put("5","Emma Esteban");

        // Enumerate over the values that were just added.
        Enumeration htEnum = ht.elements();
        while (htEnum.hasMoreElements())
        {
            System.out.println(htEnum.nextElement());
        }
    }
}

/*
Output:
Emma Esteban
Audrey Esteban
Eve Kennedy
*/
```

See Also

Concepts

Enumeration Members

java.util Package

Enumeration Members

Provides an efficient mechanism to traverse the elements of various collections.

The following tables list the members exposed by the Enumeration type.

Public Methods

Name	Description
hasMoreElements	Determines whether there are any more elements in the collection beyond where the enumerator is currently positioned.
nextElement	Advances the enumerator one step in the collection and retrieves the element at that position.

See Also

Reference

Enumeration Interface

Concepts

java.util Package

Enumeration Methods

Public Methods

Name	Description
hasMoreElements	Determines whether there are any more elements in the collection beyond where the enumerator is currently positioned.
nextElement	Advances the enumerator one step in the collection and retrieves the element at that position.

See Also

Reference

Enumeration Interface

Concepts

java.util Package

Enumeration.hasMoreElements Method

Determines whether there are any more elements in the collection beyond where the enumerator is currently positioned.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean hasMoreElements();
```

Return Value

true if there are more elements in the collection beyond where the enumerator is currently positioned; false otherwise.

Example

See the example on Enumeration.

See Also

Reference

Enumeration Interface

Concepts

Enumeration Members

java.util Package

Enumeration.nextElement Method

Advances the enumerator one step in the collection and retrieves the element at that position.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object nextElement();
```

Return Value

The element in the collection one step beyond where the enumerator was positioned before nextElement was called.

Example

See the example on Enumeration.

See Also

Reference

Enumeration Interface

Concepts

Enumeration Members

java.util Package

EventListener Interface

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.EventListener
```

See Also
Concepts
java.util Package

EventObject Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.EventObject
    extends java.lang.Object
    implements java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

java.util.EventObject

java.awt.AWTEvent

java.beans.PropertyChangeEvent

See Also

Concepts

EventObject Members

java.util Package

EventObject Members

The following tables list the members exposed by the EventObject type.

Public Constructors

Name	Description
EventObject	Overloaded.

Public Fields

Name	Description
source	

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
GetObjectData	
getSource	
getClass	(inherited from Object)
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

EventObject Class

Concepts

java.util Package

EventObject Fields

Public Fields

Name	Description
source	

See Also

Reference

EventObject Class

Concepts

java.util Package

EventObject.source Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected transient java.lang.Object source;
```

See Also

Reference

EventObject Class

Concepts

EventObject Members

java.util Package

EventObject Constructor

Overload List

Name	Description
EventObject (Object)	
EventObject (SerializationInfo, StreamingContext)	

See Also

Reference

EventObject Class

Concepts

EventObject Members

java.util Package

EventObject Constructor (Object)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.EventObject(
    java.lang.Object source);
```

Parameters

source

See Also

Reference

EventObject Class

Concepts

EventObject Members

java.util Package

EventObject Constructor (SerializationInfo, StreamingContext)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.EventObject(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also
Reference
EventObject Class
Concepts
EventObject Members
java.util Package

EventObject Methods

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
GetObjectData	
getSource	
getClass	(inherited from Object)
clone	
toString	Overridden.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

EventObject Class

Concepts

java.util Package

EventObject.GetObjectData Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

EventObject Class

Concepts

EventObject Members

java.util Package

EventObject.getSource Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object getSource();
```

See Also

Reference

EventObject Class

Concepts

EventObject Members

java.util Package

EventObject.toString Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String toString();
```

See Also

Reference

EventObject Class

Concepts

EventObject Members

java.util Package

GregorianCalendar Class

Extends the abstract class Calendar and provides the common standard calendar.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.GregorianCalendar
    extends java.util.Calendar
```

Inheritance Hierarchy

java.lang.Object

  java.util.Calendar

   java.util.GregorianCalendar

See Also

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Members

Extends the abstract class Calendar and provides the common standard calendar.

The following tables list the members exposed by the GregorianCalendar type.

Public Constructors

Name	Description
GregorianCalendar	Overloaded.

Public Fields

Name	Description
AD	Represents the ERA field, which indicates the Common Era, in a GregorianCalendar object.
areFieldsSet	Determines whether all the fields of a Calendar object are set.(inherited from Calendar)
BC	Represents the ERA field, which indicates the period before the Common Era, in a GregorianCalendar object.
fields	An array of all the fields that can be set.(inherited from Calendar)
isSet	An array of boolean values indicating which fields are set and which fields are not set.(inherited from Calendar)
isTimeSet	A value indicating whether the time is set.(inherited from Calendar)
time	A field containing the time in milliseconds.(inherited from Calendar)

Public Methods

Name	Description
add	Overridden. Adds the specified amount of time to a calendar field.
after	Overridden. Determines whether the date-time in an instance of a GregorianCalendar object is aft er the date-time in the calendar specified.
before	Overridden. Determines whether the date-time in an instance of a GregorianCalendar object is be fore the date-time in the calendar specified.
clear	Overloaded. Clears the fields of an instance of a Calendar object. (inherited from Calendar)
complete	Sets all the fields of an instance of a Calendar object. (inherited from Calendar)
computeFields	Overridden. Computes values for all the fields of an instance of a GregorianCalendar object.
computeTime	Overridden. Computes the time for a current instance of a GregorianCalendar object.
equals	Overridden. Determines whether two GregorianCalendar objects are equal.
get	Gets the value of the specified field. (inherited from Calendar)
getFirstDayOfWeek	Gets a value representing the first day of the week. (inherited from Calendar)
getGreatestMinimum	Overridden. Gets a value representing the greatest possible minimum value for a given field.

getGregorianChange	Retrieves the date of switching from Julian to Gregorian calendar.
hashCode	Overridden. Returns the hash code of a GregorianCalendar object.
getLeastMaximum	Overridden. Gets a value representing the smallest possible maximum value for a given field.
getMaximum	Overridden. Gets a value representing the maximum value for a given field.
getMinimalDaysInFirstWeek	Gets a value representing the minimal number of days for the first week of the calendar year. (inherited from Calendar)
getMinimum	Overridden. Gets a value representing the minimum value for a given field.
getTime	Gets a Date representing the time of a Calendar object. (inherited from Calendar)
getTimeInMillis	Gets the time of a Calendar object represented in milliseconds. (inherited from Calendar)
getTimeZone	Gets the TimeZone of a Calendar object. (inherited from Calendar)
getClass	(inherited from Object)
internalGet	Gets the value of a given field. (inherited from Calendar)
isLeapYear	Checks if a specified year is a leap year.
isLenient	Determines whether a given Calendar is lenient. (inherited from Calendar)
isSet	Determines whether a given field is set. (inherited from Calendar)
MemberwiseClone	(inherited from Calendar)
roll	Overridden. Modifies the value of a given field by one.
set	Overloaded. Sets various fields of a Calendar object. (inherited from Calendar)
setFirstDayOfWeek	Sets a value representing the first day of the week. (inherited from Calendar)
setGregorianChange	Sets a specified date for the GregorianCalendar calendar change.
setLenient	Sets a value that determines whether a given Calendar is lenient. (inherited from Calendar)
setMinimalDaysInFirstWeek	Sets a value representing the minimal number of days for the first week of the calendar year. (inherited from Calendar)
setTime	Sets the time of a Calendar object to the given Date. (inherited from Calendar)
setTimeInMillis	Sets the time of a Calendar object to a value represented in milliseconds. (inherited from Calendar)
setTimeZone	Sets the TimeZone of a Calendar object. (inherited from Calendar)
toString	Generates a human readable representation of a Calendar object. (inherited from Calendar)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

GregorianCalendar Class

Concepts

java.util Package

GregorianCalendar Fields

Public Fields

Name	Description
AD	Represents the ERA field, which indicates the Common Era, in a GregorianCalendar object.
areFieldsSet	Determines whether all the fields of a Calendar object are set. (inherited from Calendar)
BC	Represents the ERA field, which indicates the period before the Common Era, in a GregorianCalendar object.
fields	An array of all the fields that can be set. (inherited from Calendar)
isSet	An array of boolean values indicating which fields are set and which fields are not set. (inherited from Calendar)
isTimeSet	A value indicating whether the time is set. (inherited from Calendar)
time	A field containing the time in milliseconds. (inherited from Calendar)

See Also

Reference

GregorianCalendar Class

Concepts

java.util Package

GregorianCalendar.AD Field

Represents the ERA field, which indicates the Common Era, in a GregorianCalendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int AD;
```

Example

```
// gcal-BC_AD.jsl
// GregorianCalendar.AD-BC example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        GregorianCalendar cal = new GregorianCalendar(2004,9,10);

        // Display the contants BC AD:
        System.out.println("BC = " + cal.BC);   // 0
        System.out.println("AD = " + cal.AD);   // 1

    }
}

/*
Output:
0
1
*/
```

Remarks

The Common Era is also known as CE.

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.BC Field

Represents the ERA field, which indicates the period before the Common Era, in a GregorianCalendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final int BC;
```

Example

See the example on AD.

Remarks

The period before the Common Era is also known as BCE.

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Constructor

Overload List

Name	Description
GregorianCalendar ()	Constructs a new GregorianCalendar object.
GregorianCalendar (Locale)	Constructs a new GregorianCalendar object with a specified locale.
GregorianCalendar (TimeZone)	Constructs a new GregorianCalendar object with a specified time zone.
GregorianCalendar (TimeZone, Locale)	Constructs a new GregorianCalendar object with a specified time zone and locale.
GregorianCalendar (int, int, int)	Constructs a new GregorianCalendar object with a specified date using the default time zone and locale.
GregorianCalendar (int, int, int, int, int)	Constructs a new GregorianCalendar object with a specified date using the default time zone and locale.
GregorianCalendar (int, int, int, int, int, int)	Constructs a new GregorianCalendar object with a specified date using the default time zone and locale.

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Constructor ()

Constructs a new GregorianCalendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.GregorianCalendar();
```

Example

```
// Create a calendar object:
Calendar cal = new GregorianCalendar();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Constructor (Locale)

Constructs a new GregorianCalendar object with a specified locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.GregorianCalendar(
    java.util.Locale aLocale);
```

Parameters

aLocale

The specified locale.

Example

```
// gcal-ctor2.jsl
// GregorianCalendar #ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a locale:
        Locale loc = new Locale("en","us");
        // Create a calendar object using loc:
        Calendar cal = new GregorianCalendar(loc);
    }
}
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Constructor (TimeZone)

Constructs a new GregorianCalendar object with a specified time zone.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.GregorianCalendar(
    java.util.TimeZone zone);
```

Parameters

zone

The specified time zone.

Example

```
// gcal-ctor3.jsl
// GregorianCalendar #ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create Pacific time zone with -8 hours offset:
        TimeZone tz = new SimpleTimeZone(-28800000,
                                         "America/Los_Angeles");
         // Create a calendar object using the time zone:
         Calendar cal = new GregorianCalendar(tz);

        // Display the time zone offset:
        System.out.println("TZ offset in hours: " +
                  cal.get(Calendar.ZONE_OFFSET)/(1000*60*60));
    }
}

/*
Output:
TZ offset in hours: -8
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Constructor (TimeZone, Locale)

Constructs a new GregorianCalendar object with a specified time zone and locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.GregorianCalendar(
    java.util.TimeZone zone,
    java.util.Locale aLocale);
```

Parameters

zone

The specified time zone.

aLocale

The specified locale.

Example

```
// gcal-ctor4.jsl
// GregorianCalendar #ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create Pacific time zone with -8 hours offset:
        TimeZone tz = new SimpleTimeZone(-28800000,
                                         "America/Los_Angeles");

        // Create a locale:
        Locale loc = new Locale("en","us");

        Calendar cal = new GregorianCalendar(tz,loc);

        // Display the locale and time zone offset:
        System.out.println("Locale: " + loc);
        System.out.println("TZ offset in hours: " +
                  cal.get(Calendar.ZONE_OFFSET)/(1000*60*60));
    }
}

/*
Output:
Locale: en_US
TZ offset in hours: -8
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Constructor (Int32, Int32, Int32)

Constructs a new GregorianCalendar object with a specified date using the default time zone and locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.GregorianCalendar(
    int year,
    int month,
    int date);
```

Parameters

year

The specified year.

month

The specified month.

date

The specified day.

Example

```
// gcal-ctor5.jsl
// GregorianCalendar.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a calendar:
        Calendar cal = new GregorianCalendar(1999,8,12);

        // Get and display information from the calendar:
        System.out.println("Year: " + cal.get(Calendar.YEAR));
        System.out.println("Month: " + cal.get(Calendar.MONTH));
        System.out.println("Day: " + cal.get(Calendar.DATE));
    }
}

/*
Output:
Year: 1999
Month: 8
Day: 12
*/
```

Remarks

The values year, month and date are used to set the fields YEAR, MONTH and DATE in the calendar object.

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Constructor (Int32, Int32, Int32, Int32, Int32)

Constructs a new GregorianCalendar object with a specified date using the default time zone and locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.GregorianCalendar(
    int year,
    int month,
    int date,
    int hour,
    int minute);
```

Parameters

year

The specified year.

month

The specified month.

date

The specified day.

hour

The specified hour.

minute

The specified minute.

Example

```
// gcal-ctor5.jsl
// GregorianCalendar.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a calendar:
        Calendar cal = new GregorianCalendar(1999,8,12,6,25);

         // Get and display information from the calendar:
        System.out.println("Year: " + cal.get(Calendar.YEAR));
        System.out.println("Month: " + cal.get(Calendar.MONTH));
        System.out.println("Day: " + cal.get(Calendar.DATE));
        System.out.println("Hour: " + cal.get(Calendar.HOUR));
        System.out.println("Minute: " + cal.get(Calendar.MINUTE));
    }
}

/*
Output:
Year: 1999
Month: 8
Day: 12
Hour: 6
```

```
Minute: 25
*/
```

Remarks

The values year, month, date, hour and minute are used to set the fields YEAR, MONTH, DATE, HOUR and MINUTE in the calendar object.

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Constructor (Int32, Int32, Int32, Int32, Int32, Int32)

Constructs a new GregorianCalendar object with a specified date using the default time zone and locale.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.GregorianCalendar(
    int year,
    int month,
    int date,
    int hour,
    int minute,
    int second);
```

Parameters

year

The specified year.

month

The specified month.

date

The specified day.

hour

The specified hour.

minute

The specified minute.

second

The specified second.

Example

```
// gcal-ctor6.jsl
// GregorianCalendar.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a calendar:
        Calendar cal = new GregorianCalendar(1999,8,12,6,25,22);

        // Get and display information from the calendar:
        System.out.println("Year: " + cal.get(Calendar.YEAR));
        System.out.println("Month: " + cal.get(Calendar.MONTH));
        System.out.println("Day: " + cal.get(Calendar.DATE));
        System.out.println("Hour: " + cal.get(Calendar.HOUR));
        System.out.println("Minute: " + cal.get(Calendar.MINUTE));
        System.out.println("Second: " + cal.get(Calendar.SECOND));
    }
}
```

```
/*
Output:
Year: 1999
Month: 8
Day: 12
Hour: 6
Minute: 25
Second: 22
*/
```

Remarks

The values year, month, date, hour, minute and second are used to set the fields YEAR, MONTH, DATE, HOUR, MINUTE and SECOND in the calendar object.

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar Methods

Public Methods

Name	Description
add	Overridden. Adds the specified amount of time to a calendar field.
after	Overridden. Determines whether the date-time in an instance of a GregorianCalendar object is after the date-time in the calendar specified.
before	Overridden. Determines whether the date-time in an instance of a GregorianCalendar object is before the date-time in the calendar specified.
clear	Overloaded. Clears the fields of an instance of a Calendar object. (inherited from Calendar)
complete	Sets all the fields of an instance of a Calendar object. (inherited from Calendar)
computeFields	Overridden. Computes values for all the fields of an instance of a GregorianCalendar object.
computeTime	Overridden. Computes the time for a current instance of a GregorianCalendar object.
equals	Overridden. Determines whether two GregorianCalendar objects are equal.
get	Gets the value of the specified field. (inherited from Calendar)
getFirstDayOfWeek	Gets a value representing the first day of the week. (inherited from Calendar)
getGreatestMinimum	Overridden. Gets a value representing the greatest possible minimum value for a given field.
getGregorianChange	Retrieves the date of switching from Julian to Gregorian calendar.
hashCode	Overridden. Returns the hash code of a GregorianCalendar object.
getLeastMaximum	Overridden. Gets a value representing the smallest possible maximum value for a given field.
getMaximum	Overridden. Gets a value representing the maximum value for a given field.
getMinimalDaysInFirstWeek	Gets a value representing the minimal number of days for the first week of the calendar year. (inherited from Calendar)
getMinimum	Overridden. Gets a value representing the minimum value for a given field.
getTime	Gets a Date representing the time of a Calendar object. (inherited from Calendar)
getTimeInMillis	Gets the time of a Calendar object represented in milliseconds. (inherited from Calendar)
getTimeZone	Gets the TimeZone of a Calendar object. (inherited from Calendar)
getClass	(inherited from Object)
internalGet	Gets the value of a given field. (inherited from Calendar)
isLeapYear	Checks if a specified year is a leap year.

isLenient	Determines whether a given Calendar is lenient. (inherited from Calendar)
isSet	Determines whether a given field is set. (inherited from Calendar)
MemberwiseClone	(inherited from Calendar)
roll	Overridden. Modifies the value of a given field by one.
set	Overloaded. Sets various fields of a Calendar object. (inherited from Calendar)
setFirstDayOfWeek	Sets a value representing the first day of the week. (inherited from Calendar)
setGregorianChange	Sets a specified date for the GregorianCalendar calendar change.
setLenient	Sets a value that determines whether a given Calendar is lenient. (inherited from Calendar)
setMinimalDaysInFirstWeek	Sets a value representing the minimal number of days for the first week of the calendar year. (inherited from Calendar)
setTime	Sets the time of a Calendar object to the given Date. (inherited from Calendar)
setTimeInMillis	Sets the time of a Calendar object to a value represented in milliseconds. (inherited from Calendar)
setTimeZone	Sets the TimeZone of a Calendar object. (inherited from Calendar)
toString	Generates a human readable representation of a Calendar object. (inherited from Calendar)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

GregorianCalendar Class

Concepts

java.util Package

GregorianCalendar.add Method

Adds the specified amount of time to a calendar field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void add(
    int fld,
    int amount);
```

Parameters

fld

The field to be modified.

amount

The amount of time to add to the above field.

Example

```
// gcal-add1.jsl
// GregorianCalendar.add example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct a calendar:
        Calendar cal = new GregorianCalendar();

        // Add 4 years to the current calendar:
        cal.add(1,4);
        System.out.println("Year: " + cal.get(Calendar.YEAR));
    }
}

/*
Output:
Year: 2008
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.after Method

Determines whether the date-time in an instance of a GregorianCalendar object is after the date-time in the calendar specified.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean after(
    java.lang.Object cal);
```

Parameters

cal

The calendar to compare to the current object.

Return Value

true if the current calendar contains a later date/time than cal; false otherwise.

Example

```
// gcal-after1.jsl
// GregorianCalendar.after example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct the current calendar:
        Calendar cal1 = new GregorianCalendar();

        // Construct another calendar:
        Calendar cal2 = new GregorianCalendar(2004,7,10);

        // Check if cal1 is after cal2:
        System.out.println(cal1.after(cal2));
    }
}

/*
Output:
true
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.before Method

Determines whether the date-time in an instance of a GregorianCalendar object is before the date-time in the calendar specified.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean before(
    java.lang.Object cal);
```

Parameters

cal

The calendar to compare to the current object.

Return Value

true if the current calendar contains an earlier date-time than cal; false otherwise.

Example

```
// gcal-before1.jsl
// GregorianCalendar.after example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Construct the current calendar:
        Calendar cal1 = new GregorianCalendar();

        // Construct another calendar:
        Calendar cal2 = new GregorianCalendar(2004,7,10);

        // Check if cal1 is before cal2:
        System.out.println(cal1.before(cal2));
    }
}

/*
Output:
false
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.computeFields Method

Computes values for all the fields of an instance of a GregorianCalendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected void computeFields();
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.computeTime Method

Computes the time for a current instance of a GregorianCalendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected void computeTime();
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.equals Method

Determines whether two GregorianCalendar objects are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The object to compare to.

Return Value

true if the two objects are equal; false otherwise.

Example

```
// gcal-equals1.jsl
// GregorianCalendar.Equals example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create two calendar objects:
        Calendar cal2 = new GregorianCalendar(2004,8,10);
        Calendar cal1 = new GregorianCalendar(2004,8,10);

        // Compare the two objects:
        System.out.println(cal1.Equals(cal2));
    }
}

/*
Output:
true
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.getGreatestMinimum Method

Gets a value representing the greatest possible minimum value for a given field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getGreatestMinimum(
    int fld);
```

Parameters

fld

The field whose greatest possible minimum value is to be retrieved.

Return Value

The greatest possible minimum value for fld.

Example

```
// gcal-Min1.jsl
// GregorianCalendar.getLeastMinimum example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = new GregorianCalendar(2004,9,10);

        // Display the greatest min. for field #1:
        System.out.println("The minimum is: " +
                           cal.getGreatestMinimum(1));
    }
}

/*
Output:
The minimum is: 1
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.getGregorianChange Method

Retrieves the date of switching from Julian to Gregorian calendar.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final java.util.Date getGregorianChange();
```

Return Value

The Gregorian switching date for the current calendar.

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.hashCode Method

Returns the hash code of a GregorianCalendar object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized int hashCode();
```

Return Value

The hash code of the GregorianCalendar object.

Example

```
// gcal-GetHCode.jsl
// GregorianCalendar.GetHashCode example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = new GregorianCalendar();

        // Display the hash code:
        System.out.println(cal.GetHashCode());
    }
}

/*
Output:
12
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.getLeastMaximum Method

Gets a value representing the smallest possible maximum value for a given field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getLeastMaximum(
    int fld);
```

Parameters

fld

The field whose smallest possible maximum value is to be retrieved.

Return Value

The smallest possible maximum value for fld.

Example

```
// gcal-Max1.jsl
// GregorianCalendar.getLeastMaximum example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create the two objects:
        Calendar cal = new GregorianCalendar(2004,9,10);

        // Display the least max. for field #1:
        System.out.println("The maximum is: " +
                           cal.getLeastMaximum(1));
    }
}

/*
Output:
The maximum is: 5000000
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.getMaximum Method

Gets a value representing the maximum value for a given field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getMaximum(
    int fld);
```

Parameters

fld

The field whose maximum value is to be retrieved.

Return Value

The maximum value for fld.

Example

```
// gcal-Max2.jsl
// GregorianCalendar.getMaximum example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = new GregorianCalendar(2004,9,10);

        // Display the least max. for field #1:
        System.out.println("The maximum is: " +
                           cal.getMaximum(1));
    }
}

/*
Output:
The maximum is: 5000000
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.getMinimum Method

Gets a value representing the minimum value for a given field.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getMinimum(
    int fld);
```

Parameters

fld

The field whose minimum value is to be retrieved.

Return Value

The minimum value for fld.

Example

```
// gcal-Min2.jsl
// GregorianCalendar.getMinimum example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = new GregorianCalendar(2004,9,10);

        // Display the least max. for field #1:
        System.out.println("The minimum is: " +
                           cal.getMinimum(1));
    }
}

/*
Output:
The minimum is: 1
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.isLeapYear Method

Checks if a specified year is a leap year.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isLeapYear(
    int year);
```

Parameters

year

The year to be checked.

Return Value

true if year is a leap year; false otherwise.

Example

```
// gcal-isLYr1.jsl
// GregorianCalendar.isLeapYear example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        GregorianCalendar cal = new GregorianCalendar();

        // Check if the year 2000 is a leap year:
        System.out.println(cal.isLeapYear(2000));
    }
}

/*
Output:
true
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.roll Method

Modifies the value of a given field by one.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void roll(
    int fld,
    boolean up);
```

Parameters

fld

The field whose value is to be modified.

up

Flag indicating whether the field is to be increased or decreased. true indicates the field is to be increased.

Example

```
// gcal-roll1.jsl
// GregorianCalendar.roll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a calendar object:
        Calendar cal = new GregorianCalendar();

        // Display the year:
        System.out.println("The current year: " + cal.get(Calendar.YEAR));

        // Roll by 1:
        cal.roll(1,true);

        // Display the new year:
        System.out.println("The new year: " + cal.get(Calendar.YEAR));
    }
}

/*
Output:
The current year: 2004
The new year: 2005
*/
```

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

GregorianCalendar.setGregorianChange Method

Sets a specified date for the GregorianCalendar calendar change.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setGregorianChange(
    java.util.Date date);
```

Parameters

date

The specified change date.

Example

See Also

Reference

GregorianCalendar Class

Concepts

GregorianCalendar Members

java.util Package

HashMap Class

Represents a unique collection of key-value pairs. The elements are sorted according to the hash value of the key, and each key can exist only once in the collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.HashMap
    extends java.util.AbstractMap
    implements java.util.Map, java.lang.Cloneable, java.io.Serializable
```

Example

The following example demonstrates the containsKey, containsValue, entrySet, get, isEmpty, keySet, put, remove, size and values methods of the HashMap class.

```
// hashmap_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with three key/value pairs.
        HashMap hm = new HashMap();
        hm.put("One", "1");
        hm.put("Two", "2a");
        hm.put("Two", "2b");
        hm.put("Three", "3");

        // Iterate over the HashMap to see what we just put in.
        Set set = hm.entrySet();
        Iterator setIter = set.iterator();
        while (setIter.hasNext())
        {
            System.out.println(setIter.next());
        }

        // Print the size of the HashMap.  Hint, it won't be 4!
        if (!hm.isEmpty())
        {
            System.out.println("The HashMap contians " +
                hm.size() + " entries.");
        }
        else
        {
            System.out.println("The HashMap is empty!");
        }

        // Print out the keys and the values.
        Set keys = hm.keySet();
        Iterator keysIter = keys.iterator();
        while (keysIter.hasNext())
        {
            System.out.println("key: " + keysIter.next());
        }

        Collection values = hm.values();
        Iterator valuesIter = values.iterator();
        while (valuesIter.hasNext())
        {
```

```
            System.out.println("value: " + valuesIter.next());
        }

        // Look for a value corresponding to a key.
        if (hm.containsKey("Two"))
        {
            Object v = hm.get("Two");
            System.out.println("The value corresponding to the " +
                "key \"Two\" is " + v.toString());
        }

        // Look for the value "2a".
        if (!hm.containsValue("2a"))
        {
            System.out.println("The value \"2a\" does not exist " +
                "because it was replaced by the value \"2b\".");
        }

        // Remove an entry from the HashMap.
        if (hm.containsKey("Three"))
        {
            Object v = hm.remove("Three");
            System.out.println("The value " + v.toString() +
                " was removed from the HashMap.");
        }
    }
}

/*
Output:
[One, 1]
[Two, 2b]
[Three, 3]
The HashMap contians 3 entries.
key: One
key: Two
key: Three
value: 1
value: 2b
value: 3
The value corresponding to the key "Two" is 2b
The value "2a" does not exist because it was replaced by the value "2b".
The value 3 was removed from the HashMap.
*/
```

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractMap

  java.util.HashMap

See Also

Concepts

HashMap Members

java.util Package

HashMap Members

Represents a unique collection of key-value pairs. The elements are sorted according to the hash value of the key, and each key can exist only once in the collection.

The following tables list the members exposed by the HashMap type.

Public Constructors

Name	Description
HashMap	Overloaded. Constructs a new instance of a HashMap object.

Public Methods

Name	Description
clear	Overridden. Clears the contents of a HashMap object.
containsKey	Overridden. Determines whether a key exists in a HashMap object.
containsValue	Overridden. Determines whether a value exists in a HashMap object.
entrySet	Overridden. Creates and populates a Set object with the elements in the HashMap object.
equals	(inherited from AbstractMap)
get	Overridden. Retrieves the value associated with the given key.
hashCode	(inherited from AbstractMap)
getClass	(inherited from Object)
isEmpty	Overridden. Determines whether the HashMap object contains any elements.
keySet	Overridden. Creates and populates a Set object with all the keys in the HashMap object.
clone	Creates a shallow copy of a HashMap object.
put	Overridden. Inserts a new key-value pair into the HashMap object.
putAll	Overridden. Inserts all the key-value pairs in the Map object into the HashMap object.
remove	Overridden. Removes the key-value pair associated with the given key from the HashMap object.
size	Overridden. Determines the number of key-value pairs in the HashMap object.
toString	(inherited from AbstractMap)
values	Overridden. Creates and populates a Set object with all the values in the HashMap object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

HashMap Class

Concepts

java.util Package

HashMap Constructor

Constructs a new instance of a HashMap object.

Overload List

Name	Description
HashMap ()	Constructs a new instance of a HashMap object.
HashMap (int)	Constructs a new instance of a HashMap object with the specified size.
HashMap (Map)	Constructs a new instance of a HashMap object, and initializes it with a Map object.
HashMap (int, float)	Constructs a HashMap object with the initial size and load factor specified.

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap Constructor ()

Constructs a new instance of a HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.HashMap();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap Constructor (Int32)

Constructs a new instance of a HashMap object with the specified size.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.HashMap(
    int initCap);
```

Parameters

initCap

The initial size of the HashMap object.

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap Constructor (Map)

Constructs a new instance of a HashMap object, and initializes it with a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.HashMap(
    java.util.Map m);
```

Parameters

m

The Map object used to initialize the HashMap object.

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap Constructor (Int32, Single)

Constructs a HashMap object with the initial size and load factor specified.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.HashMap(
    int initCap,
    float loadFactor);
```

Parameters

initCap

The initial size of the HashMap object.

loadFactor

A value used to determine when the HashMap object should be rehashed. This value must be between 0.0 and 1.0.

Remarks

Specify the value of the loadFactor when you want to control when the HashMap object should be rehashed. A value of 0.7 would cause the HashMap object to be rehashed when it became 70% full.

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap Methods

Public Methods

Name	Description
clear	Overridden. Clears the contents of a HashMap object.
containsKey	Overridden. Determines whether a key exists in a HashMap object.
containsValue	Overridden. Determines whether a value exists in a HashMap object.
entrySet	Overridden. Creates and populates a Set object with the elements in the HashMap object.
equals	(inherited from AbstractMap)
get	Overridden. Retrieves the value associated with the given key.
hashCode	(inherited from AbstractMap)
getClass	(inherited from Object)
isEmpty	Overridden. Determines whether the HashMap object contains any elements.
keySet	Overridden. Creates and populates a Set object with all the keys in the HashMap object.
clone	Creates a shallow copy of a HashMap object.
put	Overridden. Inserts a new key-value pair into the HashMap object.
putAll	Overridden. Inserts all the key-value pairs in the Map object into the HashMap object.
remove	Overridden. Removes the key-value pair associated with the given key from the HashMap object.
size	Overridden. Determines the number of key-value pairs in the HashMap object.
toString	(inherited from AbstractMap)
values	Overridden. Creates and populates a Set object with all the values in the HashMap object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

HashMap Class

Concepts

java.util Package

HashMap.clear Method

Clears the contents of a HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

Example

The following example shows how to clear the contents of a HashMap object.

```
// hashmap_clear.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with three key/value pairs.
        HashMap hm = new HashMap();
        hm.put("One", new Integer(1));
        hm.put("Two", new Integer(2));
        hm.put("Three", new Integer(3));

        // Clear the key/value pairs from the HashMap.
        hm.clear();

        System.out.println("hm contains " + hm.size() +
            " elements");
    }
}

/*
Output:
hm contains 0 elements
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap.clone Method

Creates a shallow copy of a HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

The cloned HashMap object.

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap.containsKey Method

Determines whether a key exists in a HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean containsKey(
    java.lang.Object k);
```

Parameters

k

The key to check the HashMap object for.

Return Value

Returns true if the HashMap object contains the specified key; false otherwise.

Example

The following example demonstrates how to determine if a given key is present in the HashMap.

```
// hashmap_containskey.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with three key/value pairs.
        HashMap hm = new HashMap();
        hm.put("One", new Integer(1));
        hm.put("Two", new Integer(2));
        hm.put("Three", new Integer(3));

        // Does the key "Three" exist?
        boolean threeExists = hm.containsKey("Three");
        if (threeExists)
        {
            System.out.println("The key \"Three\" exists!");
        }
        else
        {
            System.out.println("The key \"Three\" does NOT exist!");
        }

        // Does the key "Seven" exist?
        boolean sevenExists = hm.containsKey("Seven");
        if (sevenExists)
        {
            System.out.println("The key \"Seven\" exists!");
        }
        else
        {
            System.out.println("The key \"Seven\" does NOT exist!");
        }
    }
}

/*
Output:
The key "Three" exists!
```

```
The key "Seven" does NOT exist!
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap.containsValue Method

Determines whether a value exists in a HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean containsValue(
    java.lang.Object v);
```

Parameters

v

The value object to check for in the HashMapobject.

Return Value

Returns true if the HashMap object contains the specified value; false otherwise.

Example

The following example demonstrates how to determine if a given value is present in the HashMap.

```
// hashmap_containsvalue.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with three key/value pairs.
        HashMap hm = new HashMap();
        hm.put("One", new Integer(1));
        hm.put("Two", new Integer(2));
        hm.put("Three", new Integer(3));

        // Does the value 3 exist?
        boolean threeExists = hm.containsValue(new Integer(3));
        if (threeExists)
        {
            System.out.println("The value 3 exists!");
        }
        else
        {
            System.out.println("The value 3 does NOT exist!");
        }

        // Does the value 7 exist?
        boolean sevenExists = hm.containsValue(new Integer(7));
        if (sevenExists)
        {
            System.out.println("The value 7 exists!");
        }
        else
        {
            System.out.println("The value 7 does NOT exist!");
        }
    }
}

/*
Output:
The value 3 exists!
```

```
The value 7 does NOT exist!
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap.entrySet Method

Creates and populates a Set object with the elements in the HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Set entrySet();
```

Return Value

A Set object containing the elements of the HashMap object.

Example

The following example populates a Set with the contents of a HashMap.

```
// hashmap_entryset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with seven entries.
        HashMap hm = new HashMap();
        hm.put("One", "1");
        hm.put("Two", "2");
        hm.put("Three", "3");
        hm.put("Four", "4");
        hm.put("Five", "5");
        hm.put("Six", "6");
        hm.put("Seven", "7");

        // Create a Set with the entries in the HashMap.
        Set set = hm.entrySet();

        // Iterate over the Set to see what it contains.
        Iterator iter = set.iterator();
        while (iter.hasNext())
        {
            Object o = iter.next();
            System.out.println("entrySet: " + o.toString());
        }
    }
}

/*
Output:
entrySet: [One, 1]
entrySet: [Two, 2]
entrySet: [Five, 5]
entrySet: [Four, 4]
entrySet: [Seven, 7]
entrySet: [Six, 6]
entrySet: [Three, 3]
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members
java.util Package

HashMap.get Method

Retrieves the value associated with the given key.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object get(
    java.lang.Object k);
```

Parameters

k

The key whose value is to be retrieved.

Return Value

The value associated with the given key.

Example

The following example shows how to get the value associated with a key.

```
// hashmap_get.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with three key/value pairs.
        HashMap hm = new HashMap();
        hm.put("One", new Integer(1));
        hm.put("Two", new Integer(2));
        hm.put("Three", new Integer(3));

        // Get the value associated with the key "Three".
        Object v = hm.get("Three");
        if (v != null)
        {
            System.out.println("The value associated with \"Three\" is " +
                v.toString());
        }
        else
        {
            System.out.println("There is no key named \"Three\" " +
                "in the HashMap.");
        }
    }
}

/*
Output:
The value associated with "Three" is 3
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap.isEmpty Method

Determines whether the HashMap object contains any elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isEmpty();
```

Return Value

Returns true if the HashMap contains any elements; false otherwise.

Example

The following example shows how to safely access a HashMap by first determining if it is empty.

```
// hashmap_isempty.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with three key/value pairs.
        HashMap hm = new HashMap();
        hm.put("One", new Integer(1));
        hm.put("Two", new Integer(2));
        hm.put("Three", new Integer(3));

        // Determine if the HashMap is empty.
        if (!hm.isEmpty())
        {
            // Do something useful with the HashMap here.
        }
    }
}
```

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap.keySet Method

Creates and populates a Set object with all the keys in the HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Set keySet();
```

Return Value

A Set object containing the keys in the HashMap object.

Example

The following example populates a Set with the keys contained in a HashMap.

```
// hashmap_keyset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with seven entries.
        HashMap hm = new HashMap();
        hm.put("One", "1");
        hm.put("Two", "2");
        hm.put("Three", "3");
        hm.put("Four", "4");
        hm.put("Five", "5");
        hm.put("Six", "6");
        hm.put("Seven", "7");

        // Create a Set with the keys in the HashMap.
        Set set = hm.keySet();

        // Iterate over the Set to see what it contains.
        Iterator iter = set.iterator();
        while (iter.hasNext())
        {
            Object o = iter.next();
            System.out.println("keySet: " + o.toString());
        }
    }
}

/*
Output:
keySet: One
keySet: Two
keySet: Five
keySet: Four
keySet: Seven
keySet: Six
keySet: Three
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members
java.util Package

HashMap.put Method

Inserts a new key-value pair into the HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object put(
    java.lang.Object k,
    java.lang.Object v);
```

Parameters

k

The key to be inserted.

v

The value to be inserted with the above key.

Return Value

The existing value associated with the given key, if it was already present in the HashMap object; null otherwise.

Example

The following example shows how to add key/value pairs to a HashMap.

```
// hashmap_put.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with three key/value pairs.
        HashMap hm = new HashMap();
        hm.put("One", new Integer(1));
        hm.put("Two", new Integer(2));
        hm.put("Three", new Integer(3));

        // Iterate over the HashMap to see what we just put in.
        Set set = hm.entrySet();
        Iterator setIter = set.iterator();
        while (setIter.hasNext())
        {
            System.out.println(setIter.next());
        }
    }
}

/*
Output:
[One, 1]
[Two, 2]
[Three, 3]
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members

HashMap.putAll Method

Inserts all the key-value pairs in the Map object into the HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void putAll(
    java.util.Map m);
```

Parameters

m

The Map object whose elements are to be inserted into the HashMap object.

Example

The following example populates a HashMap with the contents of another HashMap.

```
// hashmap_putall.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with three key/value pairs.
        HashMap hm1 = new HashMap();
        hm1.put("One", new Integer(1));
        hm1.put("Two", new Integer(2));
        hm1.put("Three", new Integer(3));

        HashMap hm2 = new HashMap();
        hm2.putAll(hm1);

        // Iterate over the HashMap to see what we just put in.
        Set set = hm2.entrySet();
        Iterator setIter = set.iterator();
        while (setIter.hasNext())
        {
            System.out.println(setIter.next());
        }
    }
}

/*
Output:
[One, 1]
[Two, 2]
[Three, 3]
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap.remove Method

Removes the key-value pair associated with the given key from the HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object remove(
    java.lang.Object k);
```

Parameters

k

The key representing the key-value pair to be removed.

Return Value

The value associated with the given key, if the key already exists in the HashMap object; null otherwise.

Example

The following example shows how to remove key/value pairs from a HashMap.

```
// hashmap_remove.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with three key/value pairs.
        HashMap hm = new HashMap();
        hm.put("One", "1");
        hm.put("Two", "2");
        hm.put("Three", "3");

        // Remove "One" and "Two" from the HashMap.
        Object o1 = hm.remove("One");
        Object o2 = hm.remove("Two");

        // Print out the values associated with "One" and "Two".
        if (o1 != null)
        {
            System.out.println("removed o1: " + o1.toString());
        }

        if (o2 != null)
        {
            System.out.println("removed o2: " + o2.toString());
        }

        System.out.println("The HashMap now contains " +
            hm.size() + " entries.");
    }
}

/*
Output:
removed o1: 1
removed o2: 2
The HashMap now contains 1 entries.
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap.size Method

Determines the number of key-value pairs in the HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

Return Value

The number of key-value pairs in the HashMap object.

Example

The following example shows how to determine the number of key/value pairs in a HashMap.

```
// hashmap_size.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with two key/value pairs. Since the key
        // "One" is added twice, only the value "2" is actually
        // present in the HashMap.
        HashMap hm = new HashMap();
        hm.put("One", "1");
        hm.put("One", "2");
        hm.put("Three", "3");

        System.out.println("The HashMap contains " +
            hm.size() + " entries.");

        // Remove "One" from the HashMap.
        Object o1 = hm.remove("One");

        System.out.println("The HashMap now contains " +
            hm.size() + " entries.");
    }
}

/*
Output:
The HashMap contains 2 entries.
The HashMap now contains 1 entries.
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members

java.util Package

HashMap.values Method

Creates and populates a Set object with all the values in the HashMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Collection values();
```

Return Value

A Set object containing the values in the HashMap object.

Example

The following example populates a Collection with the values contained in a HashMap.

```
// hashmap_values.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a HashMap with seven entries.
        HashMap hm = new HashMap();
        hm.put("One", "1");
        hm.put("Two", "2");
        hm.put("Three", "3");
        hm.put("Four", "4");
        hm.put("Five", "5");
        hm.put("Six", "6");
        hm.put("Seven", "7");

        // Create a Collection with the values in the HashMap.
        Collection coll = hm.values();

        // Iterate over the Collection to see what it contains.
        Iterator iter = coll.iterator();
        while (iter.hasNext())
        {
            Object o = iter.next();
            System.out.println("values: " + o.toString());
        }
    }
}

/*
Output:
values: 1
values: 2
values: 5
values: 4
values: 7
values: 6
values: 3
*/
```

See Also

Reference

HashMap Class

Concepts

HashMap Members
java.util Package

HashSet Class

Represents a unique collection of elements that are sorted according to their hash value, and can each exist only once in the collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.HashSet
    extends java.util.AbstractSet
    implements java.util.Set, java.lang.Cloneable, java.io.Serializable
```

Example

The following example demonstrates the add, contains, isEmpty, iterator, remove, and size methods of the HashSet class.

```
// hashset_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Populate a HashSet with three integers.
        HashSet hs = new HashSet();
        hs.add(new Integer(1));
        hs.add(new Integer(2));
        hs.add(new Integer(2));
        hs.add(new Integer(3));

        // Determine the size of the HashSet.
        System.out.println("The HashSet contains " +
            hs.size() + " elements.");

        // Iterate over the contents of the HashSet.
        Iterator iter = hs.iterator();
        while (iter.hasNext())
        {
            System.out.println(iter.next().toString());
        }

        // Determine if an element exists, and if it does,
        // remove it.
        if (!hs.isEmpty())
        {
            Integer toRemove = new Integer(2);
            if (hs.contains(toRemove))
            {
                if (hs.remove(toRemove))
                {
                    System.out.println("Successfully removed element " +
                        toRemove.toString());
                }
            }
        }
    }
}

/*
Output:
The HashSet contains 3 elements.
1
```

```
2
3
Successfully removed element 2
*/
```

Inheritance Hierarchy

java.lang.Object

java.util.AbstractCollection

java.util.AbstractSet

java.util.HashSet

See Also

Concepts

HashSet Members

java.util Package

HashSet Members

Represents a unique collection of elements that are sorted according to their hash value, and can each exist only once in the collection.

The following tables list the members exposed by the HashSet type.

Public Constructors

Name	Description
HashSet	Overloaded. Creates a new instance of a HashSet object.

Public Methods

Name	Description
add	Overridden. Adds an element to a HashSet object.
addAll	(inherited from AbstractCollection)
clear	Overridden. Clears the content of a HashSet object.
contains	Overridden. Checks if a HashSet object contains a specific element.
containsAll	(inherited from AbstractCollection)
equals	(inherited from AbstractSet)
hashCode	(inherited from AbstractSet)
getClass	(inherited from Object)
isEmpty	Overridden. Checks if the HashSet object is empty.
iterator	Overridden. Provides an iterator over the contents of the HashSet object.
clone	Creates a new instance of a HashSet object that is a shallow copy of an existing HashSet object.
remove	Overridden. Removes an element from a HashSet object.
removeAll	(inherited from AbstractCollection)
retainAll	(inherited from AbstractCollection)
size	Overridden. Returns the number of elements in the HashSet object.
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

HashSet Class

Concepts

java.util Package

HashSet Constructor

Creates a new instance of a HashSet object.

Overload List

Name	Description
HashSet ()	Constructs a HashSet object.
HashSet (Collection)	Constructs a HashSet object, and initializes it with a collection.
HashSet (int)	Constructs a HashSet object with the initial size specified.
HashSet (int, float)	Constructs a HashSet object with the initial size and load factor specified.

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet Constructor ()

Constructs a HashSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.HashSet();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet Constructor (Collection)

Constructs a HashSet object, and initializes it with a collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.HashSet(
    java.util.Collection c);
```

Parameters

c

The collection used to initialize the object.

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet Constructor (Int32)

Constructs a HashSet object with the initial size specified.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.HashSet(
    int initCap);
```

Parameters

initCap

The initial size of HashSet.

See Also
Reference
HashSet Class
Concepts
HashSet Members
java.util Package

HashSet Constructor (Int32, Single)

Constructs a HashSet object with the initial size and load factor specified.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.HashSet(
    int initCap,
    float loadFactor);
```

Parameters

initCap

The initial size of HashSet.

loadFactor

A value used to determine when HashSet should be rehashed. This value must be between 0.0 and 1.0.

Remarks

Specify the value of loadFactor when you want to control when the HashSet object should be rehashed. A value of 0.7 would cause the HashSet object to be rehashed when it became 70% full.

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet Methods

Public Methods

Name	Description
add	Overridden. Adds an element to a HashSet object.
addAll	(inherited from AbstractCollection)
clear	Overridden. Clears the content of a HashSet object.
contains	Overridden. Checks if a HashSet object contains a specific element.
containsAll	(inherited from AbstractCollection)
equals	(inherited from AbstractSet)
hashCode	(inherited from AbstractSet)
getClass	(inherited from Object)
isEmpty	Overridden. Checks if the HashSet object is empty.
iterator	Overridden. Provides an iterator over the contents of the HashSet object.
clone	Creates a new instance of a HashSet object that is a shallow copy of an existing HashSet object.
remove	Overridden. Removes an element from a HashSet object.
removeAll	(inherited from AbstractCollection)
retainAll	(inherited from AbstractCollection)
size	Overridden. Returns the number of elements in the HashSet object.
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

HashSet Class

Concepts

java.util Package

HashSet.add Method

Adds an element to a HashSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean add(
    java.lang.Object e);
```

Parameters

e

The object to be added to the HashSet object.

Return Value

Returns true if the element was successfully added; false otherwise.

Example

```
// HashSet-add1.jsl
// HashSet example
import java.util.*;

public class MyClass
{
   public static void main()
   {
      HashSet hs = new HashSet(1);
      Integer x = new Integer(22);
      Double y = new Double(2.45);

      System.out.println(hs.add(x) + ", " + hs.add(y));
      System.out.println(hs);
   }
}

/*
Output
true, true
[2.45,22]
*/
```

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet.clear Method

Clears the content of a HashSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

Example

```
// HashSet-clear.jsl
// HashSet example
import java.util.*;

public class MyClass
{
   public static void main()
   {
      HashSet hs = new HashSet(1);

      hs.add(new Integer(22));
      hs.add(new Double(2.45));

      System.out.println("Before:" + hs);
      hs.clear();
      System.out.println("After:" + hs);
   }
}

/*
Output:
Before:[2.45,22]
After:[]
*/
```

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet.clone Method

Constructs a clone of a HashSet object by creating a shallow copy of it.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

The cloned HashSet object.

Example

```
// HashSet-clone.jsl
// HashSet clone example
import java.util.*;

public class MyClass
{
   public static void main()
   {
      HashSet hs = new HashSet();
      hs.add(new Integer(22));
      hs.add(new Double(2.45));

      System.out.println("Original: " + hs);
      System.out.println("Clone: " + hs.clone());
   }
}

/*
Output:
Original: [22,2.45]
Clone: [22,2.45]
*/
```

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet.contains Method

Checks if a HashSet object contains a specific element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean contains(
    java.lang.Object e);
```

Parameters

e

The object to search for in HashSet.

Return Value

Returns true if the HashSet object contains the element; false otherwise.

Example

```
// HashSet-contains.jsl
// HashSet contains example
import java.util.*;

public class MyClass
{
   public static void main()
   {
      HashSet hs = new HashSet();
      Integer x = new Integer(22);
      Double y = new Double(2.45);

      hs.add(x);
      hs.add(y);

      System.out.println(hs.contains(x));
   }
}

/*
Output:
true
*/
```

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet.isEmpty Method

Checks if the HashSet object is empty.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isEmpty();
```

Return Value

Returns true if the HashSet object is empty; false otherwise.

Example

The following example shows how to determine if a HashSet is empty.

```
// hashset_isempty.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Populate a HashSet with three integers.
        HashSet hs = new HashSet();
        hs.add(new Integer(1));
        hs.add(new Integer(2));
        hs.add(new Integer(3));

        if (!hs.isEmpty())
        {
            // Do something useful with the HashSet here.
        }
    }
}
```

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet.iterator Method

Provides an iterator over the contents of the HashSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Iterator iterator();
```

Return Value

An iterator that can be used to traverse the contents of the HashSet object.

Example

In the following example, an iterator is created for a HashSet object, and then used to extract the elements of the HashSet object.

```
// HashSet-iterator1.jsl
// An iterator method example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      HashSet hSet = new HashSet();

      // Add some elements to the HashSet:
      hSet.add("This");
      hSet.add(" is");
      hSet.add(" a");
      hSet.add(" test.");

      // Retrieve an iterator to the hashset:
      Iterator iter = hSet.iterator();

      // Extract elements from iterator.
      // Note that the elements may not follow the order in which they
      // are added to HashSet.
      while(iter.hasNext())
      {
         System.out.print(iter.next());
         iter.remove();
      }
   }
}
/*
Output:
 isThis a test.
*/
```

Remarks

The iterator might not return elements in the same order as they were added.

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet.remove Method

Removes an element from a HashSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean remove(
    java.lang.Object e);
```

Parameters

e

The element to be removed from the HashSet object.

Return Value

Returns true if the element was successfully removed; false otherwise.

Example

The following example shows how to remove an element from a HashSet.

```
// hashset_remove.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Populate a HashSet with three integers.
        HashSet hs = new HashSet();
        hs.add(new Integer(1));
        hs.add(new Integer(2));
        hs.add(new Integer(3));

        Integer toRemove = new Integer(2);
        if (hs.contains(toRemove))
        {
            if (hs.remove(toRemove))
            {
                System.out.println("Successfully removed element " +
                    toRemove.toString());
            }
        }
    }
}

/*
Output:
Successfully removed element 2
*/
```

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

HashSet.size Method

Returns the number of elements in the HashSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

Return Value

The number of elements in the HashSet object.

Example

```
// HashSet-size.jsl
// HashSet size example
import java.util.*;
public class MyClass
{
public static void main()
{
HashSet hs = new HashSet();
hs.add(new Integer(22));
hs.add(new Double(3.45));
System.out.println("The size is: " + hs.size());
}
}
/*
Output:
The size is: 2
*/
```

See Also

Reference

HashSet Class

Concepts

HashSet Members

java.util Package

Hashtable Class

Provides an object for key entries mapped to their corresponding values. Both keys and values must be non-null objects.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.Hashtable
    extends java.util.Dictionary
    implements java.lang.Cloneable, java.util.Map, java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

  java.util.Dictionary

  java.util.Hashtable

    java.util.Properties

See Also

Concepts

Hashtable Members

java.util Package

Hashtable Members

Provides an object for key entries mapped to their corresponding values. Both keys and values must be non-null objects.

The following tables list the members exposed by the Hashtable type.

Public Constructors

Name	Description
Hashtable	Overloaded.

Public Methods

Name	Description
clear	Clears a Hashtable object.
contains	Checks for a specific element in a Hashtable object.
containsKey	Checks for a specific key in a Hashtable object.
containsValue	Checks for a specific value in a Hashtable object.
elements	Overridden. Returns an enumeration that contains the values in a Hashtable object.
entrySet	Retrieves the set of mappings contained in a Hashtable object.
equals	Overridden. Compares a specified object to an object in a Hashtable.
get	Overridden. Retrieves an element associated with a specified key in a Hashtable object.
hashCode	Overridden. Returns the hash code of a Hashtable object.
getClass	(inherited from Object)
isEmpty	Overridden. Checks if a Hashtable object is empty.
keys	Overridden. Returns an enumeration of the keys in a Hashtable object.
keySet	Returns the key set in a Hashtable object.
clone	
put	Overridden. Adds an element at a specific key entry to a Hashtable object.
putAll	Copies all the elements from a specified map to the current Hashtable object.
rehash	Rehashes the content of the Hashtable into a larger table.
remove	Overridden. Removes a key/element pair from a Hashtable object.
size	Overridden. Returns the size (number of buckets) of a Hashtable object.
toString	Overridden. Generates the string representation of a Hashtable object.

values	Returns the set of elements in a Hashtable object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Hashtable Class

Concepts

java.util Package

Hashtable Constructor

Overload List

Name	Description
Hashtable ()	Constructs a new Hashtable object with default initial capacity (11) and load factor (0.75).
Hashtable (int)	Constructs a new Hashtable object with a specified initial capacity and default load factor.
Hashtable (Map)	Constructs a new Hashtable object with the same mappings as the specified Map object.
Hashtable (int, float)	Constructs a new Hashtable object with a specified initial capacity and load factor.

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable Constructor ()

Constructs a new Hashtable object with default initial capacity (11) and load factor (0.75).

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Hashtable();
```

Example

```
// Create a new hashtable object:
Hashtable ht = new Hashtable();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable Constructor (Int32)

Constructs a new Hashtable object with a specified initial capacity and default load factor.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Hashtable(
    int initialCapacity);
```

Parameters

initialCapacity

The specified initial capacity.

Example

```
// Create a new hashtable object with an initial capacity 20:
Hashtable ht = new Hashtable(20);
```

Remarks

Throws IllegalArgumentException if the initial capacity is negative.

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable Constructor (Map)

Constructs a new Hashtable object with the same mappings as the specified Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Hashtable(
    java.util.Map m);
```

Parameters

m

The specified Map object.

Example

In this example, you create a HashMap object, which implements the Map interface, and then you create a Hashtable object using the HashMap object. Displaying the elements of both objects gives the same result.

```
// HT-ctor3.jsl
// Map.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashMap object (implements Map):
        HashMap hm = new HashMap();

        // Add some elements:
        hm.put("4","Eve Kennedy");
        hm.put("1","Isabella Abolrous");
        hm.put("2","Nick Fredette");

        // Display the values in the hashmap object:
        System.out.println("The values in the hashmap are:\n" +
                           hm.values());

        // Create a new hashtable object using the HashMap object:
        Hashtable ht = new Hashtable(hm);

        // Display the values in the hashtable object:
        System.out.println("The values in the hashtable are:\n" +
                           ht.values());
    }
}

/*
Output:
The values in the hashmap are:
[Isabella Abolrous, Nick Fredette, Eve Kennedy]
The values in the hashtable are:
[Isabella Abolrous, Nick Fredette, Eve Kennedy]
*/
```

Remarks

Throws IllegalArgumentException if the initial capacity is negative.

See Also

Reference
Hashtable Class
Concepts
Hashtable Members
java.util Package

Hashtable Constructor (Int32, Single)

Constructs a new Hashtable object with a specified initial capacity and load factor.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Hashtable(
    int initialCapacity,
    float loadFactor);
```

Parameters

initialCapacity

The specified initial capacity.

loadFactor

The specified load factor.

Example

```
// Create a new hashtable object with initial capacity 20
// and load factor 60.0:
Hashtable ht = new Hashtable(20,60F);
```

Remarks

Throws IllegalArgumentException if the initial capacity is negative.

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable Methods

Public Methods

Name	Description
clear	Clears a Hashtable object.
contains	Checks for a specific element in a Hashtable object.
containsKey	Checks for a specific key in a Hashtable object.
containsValue	Checks for a specific value in a Hashtable object.
elements	Overridden. Returns an enumeration that contains the values in a Hashtable object.
entrySet	Retrieves the set of mappings contained in a Hashtable object.
equals	Overridden. Compares a specified object to an object in a Hashtable.
get	Overridden. Retrieves an element associated with a specified key in a Hashtable object.
hashCode	Overridden. Returns the hash code of a Hashtable object.
getClass	(inherited from Object)
isEmpty	Overridden. Checks if a Hashtable object is empty.
keys	Overridden. Returns an enumeration of the keys in a Hashtable object.
keySet	Returns the key set in a Hashtable object.
clone	
put	Overridden. Adds an element at a specific key entry to a Hashtable object.
putAll	Copies all the elements from a specified map to the current Hashtable object.
rehash	Rehashes the content of the Hashtable into a larger table.
remove	Overridden. Removes a key/element pair from a Hashtable object.
size	Overridden. Returns the size (number of buckets) of a Hashtable object.
toString	Overridden. Generates the string representation of a Hashtable object.
values	Returns the set of elements in a Hashtable object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference
Hashtable Class
Concepts
java.util Package

Hashtable.clear Method

Clears a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized void clear();
```

Example

In the following example, you create and initialize a Hashtable object and then you remove all the elements and display the values. The result is an empty set.

```
// HT-clear1.jsl
// Hashtable.clear example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Emma Esteban");

        // Clear the elements:
        ht.clear();

        // Display the values:
        System.out.println("The values are: " + ht.values());
    }
}

/*
Output:
The values are: []
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.clone Method

Creates a shallow copy of a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

A shallow copy of the Hashtable object.

Example

```
// HT-clone1.jsl
// Hashtable.clone example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht1 = new Hashtable();

        // Add some elements:
        ht1.put("4","Eve Kennedy");
        ht1.put("1","Isabella Abolrous");
        ht1.put("2","Nick Fredette");

        // Clone the hashtable object and display the clone:
        System.out.println(ht1.clone());
    }
}

/*
Output:
{1=Isabella Abolrous, 2=Nick Fredette, 4=Eve Kennedy}
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.contains Method

Checks for a specific element in a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean contains(
    java.lang.Object element);
```

Parameters

element

The element to search for in the Hashtable object.

Return Value

true if the element exists; false otherwise.

Example

In this example, you create and initialize a Hashtable object and then you check for the value "Nick Fredette." The result is true.

```
// HT-cont1.jsl
// Hashtable.contains example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        // Check for a specific element:
        System.out.println(ht.contains("Nick Fredette"));
    }
}

/*
true
*/
```

Remarks

This method is identical to the method containsValue.

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.containsKey Method

Checks for a specific key in a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean containsKey(
    java.lang.Object pkey);
```

Parameters

pkey

The key to search for in the Hashtable object.

Return Value

true if the key exists; false otherwise.

Example

In this example, you create and initialize a Hashtable object and then you check for the key "3" The result is false because this key is not in the object.

```
// HT-conkey1.jsl
// Hashtable.containsKey example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        // Check for a specific key:
        System.out.println(ht.containsKey("3"));
    }
}

/*
false
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.containsValue Method

Checks for a specific value in a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean containsValue(
    java.lang.Object v);
```

Parameters

v

The value to search for in the Hashtable object.

Return Value

true if the value exists; false otherwise.

Example

In the following example, you create and initialize a Hashtable object and then you check for the value "Eve Kennedy." The result is true.

```
// HT-conValue1.jsl
// Hashtable.containsValue example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Emma Esteban");

        // Check for a specific value:
        System.out.println(ht.containsValue("Eve Kennedy"));
    }
}

/*
true
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.elements Method

Returns an enumeration that contains the values in a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.util.Enumeration elements();
```

Return Value

The enumeration that contains the values in the Hashtable object.

Example

The following example demonstrates how to use an Enumeration to enumerate over the elements in a Hashtable object.

```
// HT-Elements1.jsl
// Hashtable.elements example

import java.util.*;

class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("1","Eve Kennedy");
        ht.put("2","Nick Fredette");
        ht.put("5","Angelina Abolrous");

        // Create an enumeration from the hashtable:
        Enumeration htEnum = ht.elements();

        // Enumerate over the values:
        while (htEnum.hasMoreElements())
        {
            System.out.println(htEnum.nextElement());
        }
    }
}

/*
Output:
Angelina Abolrous
Nick Fredette
Eve Kennedy
*/
```

Remarks

Use the Enumeration class methods to get the values of the elements.

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.entrySet Method

Retrieves the set of key/value pairs contained in a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Set entrySet();
```

Return Value

The set of mappings in the Hashtable object.

Example

```
// HT-entrysm1.jsl
// Hashtable.entrySet example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("1","Sally Abolrous");
        ht.put("3","Craig Combel");
        ht.put("5","Pille Mandla");

        // Display the entry set of the object:
        System.out.println("The entry set is:\n" + ht.entrySet());
    }
}

/*
Output:
The entry set is:
[[1, Sally Abolrous], [3, Craig Combel], [5, Pille Mandla]]
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.equals Method

Compares a specified object to this Hashtable.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object c);
```

Parameters

c

Return Value

true if the object is a map and has the same key/value pairs, false otherwise.

Example

In the following example, you create and initialize a Hashtable object and then compare the element at the key "1" to the name "Isabella Abolrous". The result is true.

```
// HT-Eq1.jsl
// Hashtable.Equals example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        Map map = new TreeMap();
        map.put("4", "Eve Kennedy");
        map.put("1", "Isabella Abolrous");
        map.put("2", "Nick Fredette");

        System.out.println(ht.equals(map));
    }
}

/*
Output:
true
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.get Method

Retrieves an element associated with a specified key in a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.Object get(
    java.lang.Object key);
```

Parameters

key

The key of the element.

Return Value

The element associated with key.

Example

```
// HT-get1.jsl
// Hashtable.get example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Emma Esteban");
        ht.put("6","Nick Fredette");

       // Get the elemnt associated with the key "6":
        System.out.println(ht.get("6"));
    }
}

/*
Output:
Nick Fredette
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.hashCode Method

Returns the hash code of a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int hashCode();
```

Return Value

The hash code of the Hashtable object.

Example

```
// HT-Hcode1.jsl
// Hashtable.GetHashCode example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        // Get the hash code of the hashtable object:
        System.out.println(ht.GetHashCode());
    }
}

/*
Output:
477852222
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.isEmpty Method

Checks if a Hashtable object is empty.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isEmpty();
```

Return Value

true if empty; false otherwise.

Example

In the following example, you create and initialize a Hashtable object and then you remove all the elements and check if the object is empty. The result is true.

```
// HT-isempt1.jsl
// Hashtable.isEmpty example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Emma Esteban");

        // Clear the elements:
        ht.clear();

        // Check if the object is empty:
        System.out.println(ht.isEmpty());
    }
}

/*
Output:
true
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.keys Method

Returns an enumeration of the keys in a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.util.Enumeration keys();
```

Return Value

An enumeration of the keys in the Hashtable object.

Example

```
// HT-keys1.jsl
// Hashtable.keys example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        // Create an enumeration of the table keys:
        Enumeration htEnum = ht.keys();

        // Enumerate over the key values:
        System.out.println("The table keys are:");
        while (htEnum.hasMoreElements())
        {
            System.out.println(htEnum.nextElement());
        }
    }
}

/*
Output:
The table keys are:
4
2
1
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.keySet Method

Returns the key set in a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Set keySet();
```

Return Value

The key set in the Hashtable object.

Example

In the following example, you create and initialize a Hashtable object and then you display the set of keys in the object.

```
// HT-keyset1.jsl
// Hashtable.keySet example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Emma Esteban");

       // Display the key set:
        System.out.println(ht.keySet());
    }
}

/*
Output:
[1, 2, 4]
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.put Method

Adds an element at a specific key entry to a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.Object put(
    java.lang.Object key,
    java.lang.Object element);
```

Parameters

key

The key entry at which the element is added.

element

The element to be added.

Return Value

The element to be added to Hashtable.

Example

In the following example, you create and initialize a Hashtable object and then you display the values stored in the object.

```
// HT-put1.jsl
// Hashtable.put example

import java.util.*;

public class MyClass
{
      public static void main(String[] args)
      {
            // Create a new hashtable object:
            Hashtable ht = new Hashtable();

            // Add some elements:
            System.out.println("The current value at \"4\" is " + ht.put("4", "Eve Kennedy"
));
            // Add a key that is already present.
            System.out.println("The current value at \"4\" is " + ht.put("4", "Jack"));
      }
}

/*
Output:
The current value at “4” is null
The current value at “4” is Eve Kennedy
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.putAll Method

Copies all the elements from a specified map to the current Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized void putAll(
    java.util.Map m);
```

Parameters

m

The specified Map object.

Example

In this example, you create and initialize a TreeMap object, which implements the Map interface, and then you copy it to a Hashtable object. Displaying the element #2 in both objects gives the same result.

```
// HT-putAll1.jsl
// Hashtable.putAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object (implements Map):
        TreeMap tm = new TreeMap();
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements to tm:
        tm.put("1","Sally Abolrous");
        tm.put("2","Craig Combel");

        // Copy tm to ht:
        ht.putAll(tm);

        // Display the element #2 in both objects:
        System.out.println("The element #2 in tm is: "
                           + tm.get("2"));
        System.out.println("The element #2 in ht is: "
                           + ht.get("2"));
    }
}

/*
Output:
The element #2 in tm is: Craig Combel
The element #2 in ht is: Craig Combel
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.rehash Method

Rehashes the content of the Hashtable into a larger table.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected void rehash();
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.remove Method

Removes a key/element pair from a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.Object remove(
    java.lang.Object key);
```

Parameters

key

The key associated with the element.

Return Value

The element associated with key. If the key is not present, it returns null.

Example

In this example, you create and initialize a Hashtable object and then you remove one element from the object.

```
// HT-rem1.jsl
// Hashtable.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("3","Sally Abolrous");
        ht.put("2","Craig Combel");
        ht.put("5","Pille Mandla");

        // Remove an object:
        System.out.println("The following object has been removed:\n" +
                           ht.remove("5"));
    }
}

/*
Output:
The following object has been removed:
Pille Mandla
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.size Method

Returns the size of a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

Return Value

The size of the Hashtable object.

Example

In this example, you create and initialize a Hashtable object and then you display the size of the table.

```
// HT-size1.jsl
// Hashtable.size example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        // Display the table size:
        System.out.println("The size of the table is: " + ht.size());
    }
}

/*
Output:
The size of the table is: 3
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.toString Method

Generates the string representation of a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.String toString();
```

Return Value

The string representation of the Hashtable object.

Example

```
// HT-toStr1.jsl
// Hashtable.toString example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        // Display the table as a string:
        System.out.println(ht.toString());
    }
}

/*
Output:
{1=Isabella Abolrous, 2=Nick Fredette, 4=Eve Kennedy}
*/
```

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Hashtable.values Method

Returns the set of elements in a Hashtable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Collection values();
```

Return Value

The set of elements in the Hashtable object.

Example

```
// HT-val1.jsl
// Hashtable.values example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new hashtable object:
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("4","Eve Kennedy");
        ht.put("1","Isabella Abolrous");
        ht.put("2","Nick Fredette");

        // Display the table vlaues:
        System.out.println("The values are:\n" + ht.values());
    }
}

/*
Output:
The values are:
[Isabella Abolrous, Nick Fredette, Eve Kennedy]
*/
```

Remarks

If the table is empty, an empty set is returned.

See Also

Reference

Hashtable Class

Concepts

Hashtable Members

java.util Package

Iterator Interface

Provides an efficient mechanism to traverse the elements of various collections.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.Iterator
```

Example

In the following example, an iterator is used to iterate over HashSet and display its elements.

```
// HashSet-iterator1.jsl
// An iterator method example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      HashSet hSet = new HashSet();

      // Add some elements to the HashSet:
      hSet.add("This");
      hSet.add(" is");
      hSet.add(" a");
      hSet.add(" test.");

      // Retrieve an iterator to the hashset:
      Iterator iter = hSet.iterator();

      // Extract elements from an iterator.
      // Note that the elements may not follow the order in which
      // they are added to HashSet.
      while(iter.hasNext())
      {
         System.out.print(iter.next());
         iter.remove();
      }
   }
}

/*
Output:
 isThis a test.
*/
```

See Also

Concepts

Iterator Members

java.util Package

Iterator Members

Provides an efficient mechanism to traverse the elements of various collections.

The following tables list the members exposed by the Iterator type.

Public Methods

Name	Description
hasNext	Determines whether there are any more elements in the collection beyond where the iterator is currently positioned.
next	Advances the iterator one step in the collection, and retrieves the element at that position.
remove	Removes the element from the collection that the iterator is currently pointing to.

See Also

Reference

Iterator Interface

Concepts

java.util Package

Iterator Methods

Public Methods

Name	Description
hasNext	Determines whether there are any more elements in the collection beyond where the iterator is currently positioned.
next	Advances the iterator one step in the collection, and retrieves the element at that position.
remove	Removes the element from the collection that the iterator is currently pointing to.

See Also

Reference

Iterator Interface

Concepts

java.util Package

Iterator.hasNext Method

Determines whether there are any more elements in the collection beyond where the iterator is currently positioned.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean hasNext();
```

Return Value

Returns true if there are more elements in the collection beyond where the iterator is currently positioned; false otherwise.

Example

In the following example, an iterator is used to iterate over HashSet and displays its elements.

```
// HashSet-iterator1.jsl
// An iterator method example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      HashSet hSet = new HashSet();

      // Add some elements to the HashSet:
      hSet.add("This");
      hSet.add(" is");
      hSet.add(" a");
      hSet.add(" test.");

      // Retrieve an iterator to the hashset:
      Iterator iter = hSet.iterator();

      // Extract elements from an iterator.
      // Note that the elements may not follow the order in which
      // they are added to HashSet.
      while(iter.hasNext())
      {
         System.out.print(iter.next());
         iter.remove();
      }
   }
}

/*
Output:
 isThis a test.
*/
```

See Also

Reference

Iterator Interface

Concepts

Iterator Members

java.util Package

Iterator.next Method

Advances the iterator one step in the collection, and retrieves the element at that position.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object next();
```

Return Value

The element in the collection one step beyond where the iterator was positioned before next was called.

Example

In the following example, an iterator is used to iterate over HashSet and displays its elements.

```
// HashSet-iterator1.jsl
// An iterator method example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      HashSet hSet = new HashSet();

      // Add some elements to the HashSet:
      hSet.add("This");
      hSet.add(" is");
      hSet.add(" a");
      hSet.add(" test.");

      // Retrieve an iterator to the hashset:
      Iterator iter = hSet.iterator();

      // Extract elements from an iterator.
      // Note that the elements may not follow the order in which
      // they are added to HashSet.
      while(iter.hasNext())
      {
         System.out.print(iter.next());
         iter.remove();
      }
   }
}

/*
Output:
 isThis a test.
*/
```

See Also

Reference

Iterator Interface

Concepts

Iterator Members

java.util Package

Iterator.remove Method

Removes the element from the collection that the iterator is currently pointing to.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void remove();
```

Example

In the following example, an iterator is used to iterate over HashSet and displays its elements.

```
// HashSet-iterator1.jsl
// An iterator method example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      HashSet hSet = new HashSet();

      // Add some elements to the HashSet:
      hSet.add("This");
      hSet.add(" is");
      hSet.add(" a");
      hSet.add(" test.");

      // Retrieve an iterator to the hashset:
      Iterator iter = hSet.iterator();

      // Extract elements from an iterator.
      // Note that the elements may not follow the order in which
      // they are added to HashSet.
      while(iter.hasNext())
      {
         System.out.print(iter.next());
         iter.remove();
      }
   }
}

/*
Output:
 isThis a test.
*/
```

See Also

Reference

Iterator Interface

Concepts

Iterator Members

java.util Package

LinkedList Class

A class that represents a dynamic collection of elements. Each element consists of a data field and a link field. The link field points to the data field of the next element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.LinkedList
    extends java.util.AbstractSequentialList
    implements java.util.List, java.lang.Cloneable, java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractCollection

    java.util.AbstractList

      java.util.AbstractSequentialList

        java.util.LinkedList

See Also

Concepts

LinkedList Members

java.util Package

LinkedList Members

A class that represents a dynamic collection of elements. Each element consists of a data field and a link field. The link field points to the data field of the next element.

The following tables list the members exposed by the LinkedList type.

Public Constructors

Name	Description
LinkedList	Overloaded.

Public Fields

Name	Description
modCount	(inherited from AbstractList)

Public Methods

Name	Description
add	Overloaded. Overridden.
addAll	Overloaded. Overridden.
addFirst	Inserts an object as the first element into a LinkedList object.
addLast	Inserts an element as the last element into a LinkedList object.
clear	Overridden. Removes all the elements from a LinkedList object.
contains	Overridden. Checks if a LinkedList object contains a specific element.
containsAll	(inherited from AbstractCollection)
equals	(inherited from AbstractList)
get	Overridden. Retrieves the element at the specified index in a LinkedList object.
getFirst	Retrieves the first element in a LinkedList object.
hashCode	(inherited from AbstractList)
getLast	Retrieves the last element in a LinkedList object.
getClass	(inherited from Object)
indexOf	Overridden. Retrieves the index of a specified element in a LinkedList object.
isEmpty	(inherited from AbstractCollection)
iterator	(inherited from AbstractSequentialList)
lastIndexOf	Overridden. Retrieves the last index at which the element was last stored in a LinkedList object.

listIterator	Overloaded.
clone	
remove	Overloaded. Overridden.
removeAll	(inherited from AbstractCollection)
removeFirst	Removes the first element from a LinkedList object.
removeLast	Removes the last element from a LinkedList object.
removeRange	(inherited from AbstractList)
retainAll	(inherited from AbstractCollection)
set	Overridden. Sets a specific element at a specific index into a LinkedList object.
size	Overridden. Retrieves the number of elements in a LinkedList object.
subList	(inherited from AbstractList)
toArray	Overloaded. Overridden.
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

LinkedList Class

Concepts

java.util Package

LinkedList Fields

Public Fields

Name	Description
modCount	(inherited from AbstractList)

See Also

Reference

LinkedList Class

Concepts

java.util Package

LinkedList Constructor

Overload List

Name	Description
LinkedList ()	Constructs a LinkedList object.
LinkedList (Collection)	Constructs a LinkedList object and initializes it with a collection.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList Constructor ()

Constructs a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.LinkedList();
```

Example

```
// Create a LinkedList object:
LinkedList lList =  new LinkedList();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList Constructor (Collection)

Constructs a LinkedList object and initializes it with a collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.LinkedList(
    java.util.Collection c);
```

Parameters

c

The collection to initialize the LinkedList object.

Example

In this example, you create an ArrayList object and initialize it by adding two elements to it. And then, you create a LinkedList object and initialize it using the ArrayList object. When you display the two objects, you get two identical collections.

```
// LList-ctor2.jsl
// LinkedList.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create an ArrayList object:
        ArrayList aList = new ArrayList();

        // Add some elements to aList:
        aList.add("Craig");
        aList.add("Sally");

       // Create a LinkedList object and itialize it
       // from the aList collection:
        LinkedList lList = new LinkedList(aList);

        // Display both collections:
        System.out.println(aList);
        System.out.println(lList);
    }
}

/*
Output:
[Craig, Sally]
[Craig, Sally]
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList Methods

Public Methods

Name	Description
add	Overloaded. Overridden.
addAll	Overloaded. Overridden.
addFirst	Inserts an object as the first element into a LinkedList object.
addLast	Inserts an element as the last element into a LinkedList object.
clear	Overridden. Removes all the elements from a LinkedList object.
contains	Overridden. Checks if a LinkedList object contains a specific element.
containsAll	(inherited from AbstractCollection)
equals	(inherited from AbstractList)
get	Overridden. Retrieves the element at the specified index in a LinkedList object.
getFirst	Retrieves the first element in a LinkedList object.
hashCode	(inherited from AbstractList)
getLast	Retrieves the last element in a LinkedList object.
getClass	(inherited from Object)
indexOf	Overridden. Retrieves the index of a specified element in a LinkedList object.
isEmpty	(inherited from AbstractCollection)
iterator	(inherited from AbstractSequentialList)
lastIndexOf	Overridden. Retrieves the last index at which the element was last stored in a LinkedList object.
listIterator	Overloaded.
clone	
remove	Overloaded. Overridden.
removeAll	(inherited from AbstractCollection)
removeFirst	Removes the first element from a LinkedList object.
removeLast	Removes the last element from a LinkedList object.
removeRange	(inherited from AbstractList)

retainAll	(inherited from AbstractCollection)
set	Overridden. Sets a specific element at a specific index into a LinkedList object.
size	Overridden. Retrieves the number of elements in a LinkedList object.
subList	(inherited from AbstractList)
toArray	Overloaded. Overridden.
toString	(inherited from AbstractCollection)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

LinkedList Class

Concepts

java.util Package

LinkedList.add Method

Overload List

Name	Description
LinkedList.add (Object)	Inserts an element into a LinkedList object.
LinkedList.add (int, Object)	Inserts an element into a LinkedList object at a specified index.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.add Method (Object)

Inserts an element into a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean add(
    java.lang.Object e);
```

Parameters

e

The element to be added to LinkedList.

Return Value

true if the element was inserted successfully; false otherwise.

Example

```
// LList-add1.jsl
// LinkedList.add example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella Abolrous");
        lList.add("Angelina Abolrous");

        // Get and display the elements:
        System.out.println("The elements of the collection are:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(lList.get(i));
    }
}

/*
Output:
The elements of the collection are:
Isabella Abolrous
Angelina Abolrous
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.add Method (Int32, Object)

Inserts an element into a LinkedList object at a specified index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void add(
    int ix,
    java.lang.Object e);
```

Parameters

ix

The index at which the element is inserted into LinkedList.

e

The element to be inserted into LinkedList.

Example

```
// LList-add2.jsl
// LinkedList.add example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Insert more elements:
        lList.add(1,"Bianca");
        lList.add(3,"Sally");

        // Get and display the elements:
        System.out.println("The elements of the collection are: ");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + lList.get(i));
    }
}

/*
Output:
The elements of the collection are:
0 = Isabella
1 = Bianca
2 = Angelina
3 = Sally
*/
```

Remarks

Throws IndexOutOfBoundsException if ix is negative, or greater than or equal to size.

See Also

Reference

LinkedList Class
Concepts
LinkedList Members
java.util Package

LinkedList.addAll Method

Overload List

Name	Description
LinkedList.addAll (Collection)	Inserts all the elements from a collection into a LinkedList object.
LinkedList.addAll (int, Collection)	Inserts all the elements from a collection into a LinkedList object starting at a specified index.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.addAll Method (Collection)

Inserts all the elements from a collection into a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean addAll(
    java.util.Collection c);
```

Parameters

c

The name of the collection.

Return Value

true if the elements were inserted successfully; false otherwise.

Example

```
// LList-addAll1.jsl
// LinkedList.addAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create ArrayList and LinkedList objects:
        ArrayList aList = new ArrayList();
        LinkedList lList = new LinkedList();

        // Initialize aList:
        aList.add("Craig");
        aList.add("Sally");

       // Add all the elements of aList to lList:
        lList.addAll(aList);

        // Display lList:
        System.out.println(lList);
    }
}

/*
Output:
[Craig, Sally]
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.addAll Method (Int32, Collection)

Inserts all the elements from a collection into a LinkedList object starting at a specified index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean addAll(
    int ix,
    java.util.Collection c);
```

Parameters

ix

The index at which the insertion starts.

c

The name of the collection.

Return Value

true if the elements were inserted successfully; false otherwise.

Example

```
// LList-addAll2.jsl
// LinkedList.addAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create an ArrayList and a LinkedList objects:
        ArrayList aList = new ArrayList();
        LinkedList lList = new LinkedList();

        // Initialize aList:
        aList.add("Craig");
        aList.add("Sally");

        // Initialize lList:
        lList.add("Pille");
        lList.add("Hazem");

       // Add all the elements of aList to lList starting at index 1:
        lList.addAll(1,aList);

        // Display lList:
        System.out.println(lList);
    }
}

/*
Output:
[Pille, Craig, Sally, Hazem]
*/
```

Remarks

Throws IndexOutOfBoundsException if ix is negative, or greater than or equal to size.

See Also

Reference
LinkedList Class
Concepts
LinkedList Members
java.util Package

LinkedList.addFirst Method

Inserts an object as the first element into a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void addFirst(
    java.lang.Object e);
```

Parameters

e

The object to be inserted.

Example

In this example, you create a linked list object, add two elements (first and last) to it, and then you print the elements.

```
// LinkedList-addFirst.jsl
// LinkedList.addFirst example

import java.util.*;

public class MyClass
{
   public static void main(String[] args)
   {
      LinkedList ll = new LinkedList();
      Integer x = new Integer(123);
      Double y = new Double(4.56);

      // Build the list with the first and last elements:
      ll.addFirst(x);
      ll.addLast(y);

      // Print the list:
      System.out.println("The list elements are: " + ll);
   }
}

/*
Output:
The list elements are: [123, 4.56]
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.addLast Method

Inserts an element as the last element into a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void addLast(
    java.lang.Object e);
```

Parameters

e

The element to be inserted.

Example

See the example under addFirst.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.clear Method

Removes all the elements from a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

Example

In this example, you create a linked list object, add two elements to it, and then clear the list. You also print the elements and the size of the list before and after clearing the list.

```
// LinkedList-clear.jsl
// LinkedList.clear example

import java.util.*;

public class MyClass
{
   public static void main(String[] args)
   {
      LinkedList ll = new LinkedList();
      Integer x = new Integer(123);
      Double y = new Double(4.56);

      // Build the list with the first and last elements:
      ll.addFirst(x);
      ll.addLast(y);

      // Print the list:
      System.out.println("The list elements are: " +
                         ll.getFirst() + ", " + ll.getLast());
      // Print the size:
      System.out.println("The size of the list is: "
                         + ll.size());

      // Clear the list:
      ll.clear();
      // Print the new size:
      System.out.println("The size after clearing the list is: "
                          + ll.size());
   }
}

/*
Output:
The list elements are: 123, 4.56
The size of the list is: 2
The size after clearing the list is: 0
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.clone Method

Creates a shallow copy of a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

The clone object.

Example

In this example, you create a linked list object, add two elements to it, clone it, and then you display the clone.

```
// LinkedList-clone.jsl
// LinkedList.Clone example

import java.util.*;

public class MyClass
{
   public static void main(String[] args)
   {
      LinkedList ll = new LinkedList();
      Integer x = new Integer(123);
      Double y = new Double(4.56);

      // Build the list with the first and last elements:
      ll.addFirst(x);
      ll.addLast(y);

      // Clone the list:
      Object cl = ll.clone();

      // Print the list clone:
      System.out.println("The clone is: " + cl);
   }
}

/*
Output:
The clone is: [123, 4.56]
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.contains Method

Checks if a LinkedList object contains a specific element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean contains(
    java.lang.Object e);
```

Parameters

e

The element searched for.

Return Value

true if the element was found, false otherwise.

Example

In this example, you initialize a linked list with two elements, and then you test the list to see if one of the elements exits in it.

```
// LList-contains1.jsl
// LinkedList.contains example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Check if the element "Angelina" exists in the list:
        if(lList.contains("Angelina"))
            System.out.println("The element exists in the list.");
    }
}

/*
Output:
The element exists in the list.
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.get Method

Retrieves the element at the specified index in a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object get(
    int ix);
```

Parameters

ix

The index of the retrieved element.

Return Value

The element at the specified index in the LinkedList object.

Example

```
// LList-get1.jsl
// LinkedList.get example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Display the element #1:
        System.out.println("The element #1 is: " + lList.get(1));
    }
}

/*
Output:
The element #1 is: Angelina
*/
```

Remarks

Throws IndexOutOfBoundsException if ix is negative, or greater than or equal to size.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.getFirst Method

Retrieves the first element in a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object getFirst();
```

Return Value

The object stored in the first element.

Example

```
// LList-getFirst1.jsl
// LinkedList.getFirst example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Display the first element:
        System.out.println("The first element is: " +
                           lList.getFirst());
    }
}

/*
Output:
The first element is: Isabella
*/
```

Remarks

Throws NoSuchElementException if the list is null.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.getLast Method

Retrieves the last element in a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object getLast();
```

Return Value

The object stored in the last element.

Example

```
// LList-getLast1.jsl
// LinkedList.getLast example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Display the first element:
        System.out.println("The last element is: " +
                           lList.getLast());
    }
}

/*
Output:
The last element is: Angelina
*/
```

Remarks

Throws NoSuchElementException if the list is null.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.indexOf Method

Retrieves the index of a specified element in a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int indexOf(
    java.lang.Object e);
```

Parameters

e

The element for which the index is retrieved,

Return Value

The index of the specified element.

Example

See the example under lastIndexOf.

Remarks

Returns -1 if the list does not contain the specified element.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.lastIndexOf Method

Retrieves the last index at which the element was last stored in a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int lastIndexOf(
    java.lang.Object e);
```

Parameters

e

The element for which the index is retrieved.

Return Value

The index at which the specified element was last stored in a LinkedList object.

Example

```
// LList-LindexOf.jsl
// LinkedList.lastIndexOf example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        LinkedList ll =  new LinkedList();
        Integer x = new Integer(123);
        Double y = new Double(4.56);

        // Add elements:
        ll.addLast(y);
        ll.addFirst(x);
        ll.addLast(y);

        // Print the indexes of y:
        System.out.println("The index of y is: " +
                           ll.indexOf(y) );
        System.out.println("The last index of y is: " +
                           ll.lastIndexOf(y) );
    }
}

/*
Output:
The index of y is: 1
The last index of y is: 2
*/
```

Remarks

Returns -1 if the list does not contain the specified element.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.listIterator Method

Overload List

Name	Description
LinkedList.listIterator ()	
LinkedList.listIterator (int)	Retrieves an iterator over a LinkedList object starting at a specified position.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.listIterator Method (Int32)

Retrieves an iterator over a LinkedList object starting at a specified position.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ListIterator listIterator(
    int ix);
```

Parameters

ix

The initial position of the iterator.

Return Value

An iterator of the LinkedList object.

Example

```
// LList-lIter1.jsl
// LinkedList.listIteratort example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Hazem");
        lList.add("Sally");
        lList.add("Sam");
        lList.add("Camelia");


        // Get a list iterator:
        ListIterator li = lList.listIterator();

      // Extract elements from the list using the list iterator.
      // Note that the elements may not follow the order in which they
      // are added to list.
        while(li.hasNext())
        {
             System.out.println(li.next());
                li.remove();
        }
    }
}

/*
Output:
Hazem
Sally
Sam
Camelia
*/
```

See Also

Reference

LinkedList Class
Concepts
LinkedList Members
java.util Package

LinkedList.remove Method

Overload List

Name	Description
LinkedList.remove (int)	Removes an element at a specified index from a LinkedList.
LinkedList.remove (Object)	Removes a specific element from a LinkedList object.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.remove Method (Int32)

Removes an element at a specified index from a LinkedList.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object remove(
    int ix);
```

Parameters

ix

The index at which the element is removed.

Return Value

The element to be removed from the LinkedList object.

Example

```
// LList-rem1.jsl
// LinkedList.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Anabella");

        // Display the elements:
        System.out.println("The old list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + lList.get(i));

        // Remove the element #3:
           System.out.println(
                  "\nThe following element has been removed: "
                  + lList.remove(3));

        // Display the new elements:
        System.out.println("\nThe new list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + lList.get(i));
    }
}

/*
Output:
The old list:
0 = Isabella
1 = Angelina
2 = Pille
3 = Anabella
```

```
The following element has been removed: Anabella

The new list:
0 = Isabella
1 = Angelina
2 = Pille
*/
```

Remarks

Throws IndexOutOfBoundsException if ix is negative, or greater than or equal to the size of the target list.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.remove Method (Object)

Removes a specific element from a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean remove(
    java.lang.Object e);
```

Parameters

e

The element to be removed.

Return Value

true if the element was removed successfully; false otherwise.

Example

```
// LList-rem2.jsl
// LinkedList.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Anabella");

        // Display the elements:
        System.out.print("The old list: "+ lList);

        // Remove the element #3:
        lList.remove("Anabella");
        System.out.println(
             "\nThe element \"Anabella\" has been removed.");

        // Display the new elements:
        System.out.print("The new list: " + lList);
    }
}

/*
Output:
The old list: [Isabella, Angelina, Pille, Anabella]
The element "Anabella" has been removed.
The new list: [Isabella, Angelina, Pille]
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

LinkedList.removeFirst Method

Removes the first element from a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object removeFirst();
```

Return Value

The element to be removed from the LinkedList object.

Example

```
// LList-rem3.jsl
// LinkedList.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Anabella");

        // Display the elements:
        System.out.println("The old list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + lList.get(i));

        // Remove the first element:
           System.out.println(
                  "\nThe following element has been removed: "
                  + lList.removeFirst());

        // Display the new elements:
        System.out.println("\nThe new list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + lList.get(i));
    }
}

/*
The old list:
0 = Isabella
1 = Angelina
2 = Pille
3 = Anabella

The following element has been removed: Isabella

The new list:
0 = Angelina
1 = Pille
2 = Anabella
*/
```

Remarks

Throws NoSuchElementException if the list is null.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.removeLast Method

Removes the last element from a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object removeLast();
```

Return Value

The element to be removed from the LinkedList object.

Example

```
// LList-rem4.jsl
// LinkedList.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Anabella");

        // Display the elements:
        System.out.println("The old list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + lList.get(i));

        // Remove the last element:
           System.out.println(
                  "\nThe following element has been removed: "
                  + lList.removeLast());

        // Display the new elements:
        System.out.println("\nThe new list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + lList.get(i));
    }
}

/*
Output:
The old list:
0 = Isabella
1 = Angelina
2 = Pille
3 = Anabella

The following element has been removed: Anabella

The new list:
0 = Isabella
1 = Angelina
2 = Pille
```

```
*/
```

Remarks

Throws NoSuchElementException if the list is null.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.set Method

Sets a specific element at a specific index into a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object set(
    int ix,
    java.lang.Object e);
```

Parameters

ix

The index of the element to be set.

e

The element to be set.

Return Value

The element to be set into the LinkedList object.

Example

```
// LList-set1.jsl
// LinkedList.set example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Hazem");

        // Display the elements:
        System.out.println("The old list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + "=" + lList.get(i));

        // Set the element #3:
           System.out.println("\nThe following element has been set: "
                              + lList.set(3,"Anabella"));

        // Display the elements after changing #3:
        System.out.println("\nThe new list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + "=" + lList.get(i));
    }
}

/*
Output:
The old list:
0=Isabella
1=Angelina
```

```
2=Pille
3=Hazem

The following element has been set: Hazem

The new list:
0=Isabella
1=Angelina
2=Pille
3=Anabella
*/
```

Remarks

Throws IndexOutOfBoundsException if ix is negative, or greater than or equal to the size of the target list.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.size Method

Retrieves the number of elements in a LinkedList object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

Return Value

The number of elements in the LinkedList object.

Example

```
// LList-size1.jsl
// LinkedList.size example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Hazem");

        // Display the size of the list:
        System.out.println("The size of the list is: " + lList.size());
    }
}

/*
Output:
The size of the list is: 4
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.toArray Method

Overload List

Name	Description
LinkedList.toArray ()	Copies values of a LinkedList object to an array.
LinkedList.toArray (Object[])	Copies the elements of a LinkedList object into specified array object.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.toArray Method ()

Copies values of a LinkedList object to an array.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object[] toArray();
```

Return Value

The array object.

Example

```
// LList-toArr1.jsl
// LinkedList.toArray example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Hazem");

        // Create an array from the list:
        Object[] s = lList.toArray();

        // Display the array elements:
        System.out.println("The array elements are:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + s[i]);
    }
}

/*
Output:
The array elements are:
0 = Isabella
1 = Angelina
2 = Pille
3 = Hazem
*/
```

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

LinkedList.toArray Method (Object[ ])

Copies the elements of a LinkedList object into specified array object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object[] toArray(
    java.lang.Object[] arr);
```

Parameters

arr

The array object that receives the elements of the LinkedList object.

Return Value

The array object that contains the LinkedList elements.

Example

In this example, you create an initialize a LinkedList collection, convert it to an array object "s," and then you copy this object to a new array object "s1."

```
// LList-toArr2.jsl
// LinkedList.toArray example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Hazem");

        // Create an array "s" from the list:
        Object[] s = lList.toArray();

        // Copy "s" to another object array "s1":
        Object[] s1 = lList.toArray(s);

        // Display the array objects:
        System.out.println("The array objects are:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + " = " + s1[i]);
    }
}

/*
Output:
The array objects are:
0 = Isabella
1 = Angelina
2 = Pille
3 = Hazem
*/
```

Remarks

Throws NullPointerException if the specified array is null.

See Also

Reference

LinkedList Class

Concepts

LinkedList Members

java.util Package

List Interface

An interface that represents an ordered collection of elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.List
    extends java.util.Collection
```

Remarks

This interface is implemented by collection classes such as ArrayList and LinkedList.

See Also

Concepts

List Members

java.util Package

List Members (J#)

An interface that represents an ordered collection of elements.

The following tables list the members exposed by the List type.

Public Methods

Name	Description
add	Overloaded.
addAll	Overloaded.
clear	Removes all the elements from a List object.
contains	Checks if a List object contains a specific element.
containsAll	Checks if a List object contains all the elements of a collection.
equals	Checks if a List object is equal to a specific object.
get	Retrieves an element from a List at a specified index.
hashCode	Retrieves the hash code of a List.object.
indexOf	Retrieves the index of a specific element in a List.object.
isEmpty	Checks if a java.util.List object is empty.
iterator	Provides an iterator to iterate over a List object.
lastIndexOf	Retrieves the last index a specific element has possessed when it was last added to List object.
listIterator	Overloaded.
remove	Overloaded.
removeAll	Deletes all the elements from a List object that exist is a specified collection.
retainAll	Deletes the elements of the current List collection that do not exist in specified collection object.
set	Changes the contents of an existing element at the specified index in a List object.
size	Retrieves the number of elements in a List object.
subList	Creates a sub list from the elements that exist within a specified index range in a List object.
toArray	Overloaded.

See Also

Reference

List Interface

Concepts

java.util Package

List Methods (J#)

Public Methods

Name	Description
add	Overloaded.
addAll	Overloaded.
clear	Removes all the elements from a List object.
contains	Checks if a List object contains a specific element.
containsAll	Checks if a List object contains all the elements of a collection.
equals	Checks if a List object is equal to a specific object.
get	Retrieves an element from a List at a specified index.
hashCode	Retrieves the hash code of a List.object.
indexOf	Retrieves the index of a specific element in a List.object.
isEmpty	Checks if a java.util.List object is empty.
iterator	Provides an iterator to iterate over a List object.
lastIndexOf	Retrieves the last index a specific element has possessed when it was last added to List object.
listIterator	Overloaded.
remove	Overloaded.
removeAll	Deletes all the elements from a List object that exist is a specified collection.
retainAll	Deletes the elements of the current List collection that do not exist in specified collection object.
set	Changes the contents of an existing element at the specified index in a List object.
size	Retrieves the number of elements in a List object.
subList	Creates a sub list from the elements that exist within a specified index range in a List object.
toArray	Overloaded.

See Also

Reference

List Interface

Concepts

java.util Package

List.add Method

Overload List

Name	Description
List.add (Object)	Inserts a specific element into a List.object.
List.add (int, Object)	Inserts a specific element at a specific index into a List object.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.add Method (Object)

Inserts the specified element into a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean add(
    java.lang.Object e);
```

Parameters

e

The element to be inserted into the List object.

Return Value

true if the element is inserted successfully; false otherwise.

Example

In this example, you create a linked list object, which implements the List interface through one of its super classes. Then you add some elements to the list and display them.

```
// list-add1.jsl
// List.add example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object (which implements the
        // List interface):
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My Node");
        lList.add("Your Node");
        lList.add(null);
        lList.add(new Integer(55));

        // Get and display the elements:
        for (int i=0; i<lList.size(); i++)
            System.out.println(lList.get(i));
    }
}

/*
Output:
My Node
Your Node
null
55
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.add Method (Int32, Object)

Inserts the specified element at the specified index into a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void add(
    int ix,
    java.lang.Object e);
```

Parameters

ix

The index at which the element is added.

e

The element to be added to the List object.

Example

```
// list-add2.jsl
// List.add example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object (which implements the
        // List interface):
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("John");
        lList.add("Sam");

        // Insert more elements:
        lList.add(0,"Fred");
        lList.add(1,"Michelle");
        lList.add(3,"Virginia");

        // Get and display the elements:
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + "=" + lList.get(i));
    }
}

/*
0=My Node
0=Fred
1=Michelle
2=John
3=Virginia
4=Sam
*/
```

Remarks

Throws IndexOutOfBoundsException if ix is negative or greater than or equal to size.

See Also

Reference
List Interface
Concepts
List Members
java.util Package

List.addAll Method

Overload List

Name	Description
List.addAll (Collection)	Inserts all the elements from a collection into a List object.
List.addAll (int, Collection)	Inserts all the elements from a collection into a List object starting at a specified index.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.addAll Method (Collection)

Inserts all the elements from a Collection into a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean addAll(
    java.util.Collection c);
```

Parameters

c

The Collection whose elements are inserted into the List object.

Return Value

true if the elements were inserted successfully; false otherwise.

Example

In this example, you declare an ArrayList, al, object and a LinkedList object, ll. You then add some elements to al and copy it to ll. When you display ll, you get the same elements as those of al.

```
// list-addAll1.jsl
// List.addAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create an ArrayList object and a LinkedList object:
        ArrayList  al =  new ArrayList();
        LinkedList ll =  new LinkedList();

        // Add some elements to al:
        al.add("My Node");
        al.add("Your Node");
        al.add(null);
        al.add(new Integer(55));

        // Copy al to ll and display ll:
        if(ll.addAll(al)==true)
            System.out.println(ll);
    }
}

/*
Output:
[My Node, Your Node, null, 55]
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.addAll Method (Int32, Collection)

Inserts all the elements from a collection into a List object starting at a specified index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean addAll(
    int ix,
    java.util.Collection c);
```

Parameters

ix

The starting index at which the elements are inserted.

c

The name of the collection.

Return Value

true if the elements are inserted successfully; false otherwise.

Example

In this example, you declare an ArrayList, al, object and a LinkedList object, ll. You the add some elements to both al and ll. Then you copy al to ll starting art index 2. When you display ll, you see that it includes both elements of al and ll.

```
// list-addAll2.jsl
// List.addAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create an ArrayList object and a LinkedList object:
        ArrayList  al =  new ArrayList();
        LinkedList ll =  new LinkedList();

        // Add some elements to al:
        al.add("My Node");
        al.add("Your Node");
        al.add(null);
        al.add(new Integer(55));

        // Add some elements to ll:
        ll.add("01");
        ll.add("02");
        ll.add("03");
        ll.add("04");

        // Copy al to ll starting at index 2:
        ll.addAll(2,al);

        // Display al and ll:
        System.out.println("al = " + al);
        System.out.println("ll = " + ll);
    }
}

/*
Output:
```

```
  al = [My Node, Your Node, null, 55]
  ll = [01, 02, My Node, Your Node, null, 55, 03, 04]
  */
```

Remarks

Throws IndexOutOfBoundsException if ix is negative, or greater than or equal to size.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.clear Method

Removes all the elements from a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void clear();
```

Example

```
// list-clear.jsl
// List example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("100");
        lList.add("200");
        lList.add("300");
        lList.add("400");

        // Get and display the elements:
        System.out.println("The elements of the list are:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(lList.get(i));

        // Print the size:
        System.out.println("The size of the list is: " + lList.size());

        // Clear the list:
        lList.clear();
        System.out.print("The list has been cleared. ");

        // Print the new size:
        System.out.println("The size is now: " + lList.size());
    }
}

/*
Output:
The elements of the list are:
100
200
300
400
The size of the list is: 4
The list has been cleared. The size is now: 0
*/
```

See Also

Reference

List Interface

Concepts

List Members

List.contains Method

Checks if a List object contains a specific element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean contains(
    java.lang.Object e);
```

Parameters

e

The element searched for in the List object.

Return Value

true if the element was found; false otherwise.

Example

In this example, you create and initialize a linked list object, and then you check to see if the list contains a specified element.

```
// list-cont1.jsl
// List.contains example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My Node");
        lList.add("Your Node");
        lList.add(null);
        lList.add(new Integer(55));

        // Check if "My Node" is in the list:
        if(lList.contains("My Node"))
            System.out.println(
                       "The object \"My Node\" is in the list.");
    }
}

/*
Output:
The object "My Node" is in the list.
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.containsAll Method

Checks if a List object contains all the elements of a collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean containsAll(
    java.util.Collection c);
```

Parameters

c

The name of the collection.

Return Value

true if all the elements were found; false otherwise.

Example

In this example, you create a linked list object, lList, and add some elements to it. Then you create another linked list object, lList1, and copy lList to it. Testing lList1 to see if it contains all the elements of lList gives the result true.

```
// list-contAll1.jsl
// List.containsAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My Node");
        lList.add("Your Node");
        lList.add(null);
        lList.add(new Integer(55));

        // Create a new list lList1:
        LinkedList lList1 =  new LinkedList();

        // Add all the elements of lList to lList1:
        lList1.addAll(lList);

        // Confirm that the elements of IList exist in IList1:
        if(lList1.containsAll(lList))
            System.out.println(
                  "All the elements exist in the new list.");
    }
}

/*
Output:
All the elements exist in the new list.
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.equals Method

Checks if a List object is equal to the specified object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean equals(
    java.lang.Object c);
```

Parameters

c

The object to compare to.

Return Value

true if the specified object is equal to this object; false otherwise.

Example

In this example, you create two linked list objects, add some elements to the first one, and copy it to the second one. Then you test the two objects for equality.

```
// list-eq1.jsl
// List.Equals example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My Node");
        lList.add("Your Node");
        lList.add(null);
        lList.add(new Integer(55));

        // Create a new list lList1:
        LinkedList lList1 =  new LinkedList();

        // Add all the element of lList to lList1:
        lList1.addAll(lList);

        // Check if the two lists are identical:
        if(lList1.Equals(lList))
            System.out.println("The two lists are the same.");
    }
}

/*
Output:
The two lists are the same.
*/
```

See Also

Reference

List Interface

Concepts

List Members
java.util Package

List.get Method

Retrieves an element from a List at a specified index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object get(
    int ix);
```

Parameters

ix

The index of the element to be read.

Return Value

The element to be retrieved from the List object.

Example

See the examples under clear and add.

Remarks

Throws IndexOutOfBoundsException if ix is negative, or greater than or equal to size.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.hashCode Method

Retrieves the hash code of a List.object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int hashCode();
```

Return Value

An integer representing the hash code of the List object.

Example

In this example, you display the hash code of a linked list object after adding some elements to it.

```
// list-getHash1.jsl
// List.GetHashCode example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add(new Double(5.25));
        lList.add(new Byte((byte)127));
        lList.add(null);
        lList.add(new Integer(55));

        // Print the hash code:
        System.out.println("The hash code is: " + lList.GetHashCode());
    }
}

/*
The hash code is: -1898639464
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.indexOf Method

Retrieves the index of a specific element in a List.object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int indexOf(
    java.lang.Object e);
```

Parameters

e

The elements for which the index is retrieved.

Return Value

The index of the element.

Example

```
// list-idxOf1.jsl
// List.indexOf example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My node");
        lList.add("Her node");
        lList.add("Their node");
        lList.add("Your node");

        // Print the index of "Your node":
        System.out.println("The index is: " +
                           lList.indexOf("Your node"));
    }
}

/*
The index is: 3
*/
```

Remarks

If the element is not found it returns -1.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.isEmpty Method

Checks if a List object is empty.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isEmpty();
```

Return Value

true if the List object is empty; false otherwise.

Example

In this example, you create and initialize a linked list object, clear it, and then you check to see if the list is empty.

```
// list-isEmp1.jsl
// List.isEmpty example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My node");
        lList.add("Her node");
        lList.add("Their node");
        lList.add("Your node");

        // Clear the list:
        lList.clear();

        // Check if the list is empty:
        if(lList.isEmpty())
            System.out.println("The list is now empty.");
    }
}

/*
The list is now empty.
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.iterator Method

Provides an iterator to iterate over a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Iterator iterator();
```

Return Value

An iterator that can be used over the List object.

Example

In this example, you create an initialize a collection and then you retrieve an iterator to iterate over the collection and display its members.

```
// list-iterator1.jsl
// List.iterator example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      Collection hSet = new HashSet();

      // Add some elements to the HashSet:
      hSet.add("1 ");
      hSet.add("2 ");
      hSet.add("3 ");
      hSet.add("4 ");

      // Retrieve an iterator to the hashset:
      Iterator iter = hSet.iterator();

      // Extract elements from iterator.
      // Note that the elements may not follow the order in which they
      // are added to HashSet.
      while(iter.hasNext())
      {
         System.out.print(iter.next());
         iter.remove();
      }
   }
}
/*
Output:
3 1 4 2
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.lastIndexOf Method

Retrieves the last index of the specified element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int lastIndexOf(
    java.lang.Object e);
```

Parameters

e

The element whose last index is retrieved.

Return Value

The last index of the element.

Example

```
// list-lastidx1.jsl
// List.lastIndexOf example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My node");
        lList.add("Her node");
        lList.add("Their node");
        lList.add("Your node");
        lList.add("Her node");

        // Print the last index of "Her node":
        System.out.println("The lst index is: " +
                           lList.lastIndexOf("Her node"));
    }
}

/*
The lst index is: 4
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.listIterator Method

Overload List

Name	Description
List.listIterator ()	Provides an iterator to traverse the elements of a List object.
List.listIterator (int)	Provides an iterator for a List object that starts at specific index position.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.listIterator Method ()

Provides an iterator to traverse the elements of a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.ListIterator listIterator();
```

Return Value

An iterator that can be used to traverse the elements of the List object.

Example

```
// List-lIter1.jsl
// List.listIteratort example

import java.util.*;

public class MyClass
{
      public static void main(String[] args)
      {
            // Create a linked list object:
            LinkedList lList = new LinkedList();

            // Add some elements:
            lList.add("Isabella");
            lList.add("Angelina");
            lList.add("Jacqueline");

            // Get a list iterator:
            ListIterator iter = lList.listIterator();

            // Extract elements from iterator:
            while (iter.hasNext())
            {
                  System.out.print(iter.next() + " ");
            }
            System.out.println();

            // Extract elements from iterator in reverse order:
            while (iter.hasPrevious())
            {
                  System.out.print(iter.previous() + " ");
            }
            System.out.println();
      }
}

/*
Output:
Isabella Angelina Jacqueline
Jacqueline Angelina Isabella
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.listIterator Method (Int32)

Provides an iterator for a List object that starts at specific index position.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.ListIterator listIterator(
    int ix);
```

Parameters

ix

The starting position of the iterator.

Return Value

The List iterator.

Example

```
// List-lIter2.jsl
// List.listIteratort example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Get a list iterator:
        ListIterator li = lList.listIterator(1);

        // Display the list iterator:
        System.out.println("The list iterator is: " + li);
    }
}

/*
Output:
The list iterator is: java.util.LinkedList$LinkIterator@33c0d9d
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.remove Method

Overload List

Name	Description
List.remove (int)	Deletes an element at a specific index from a List object.
List.remove (Object)	Deletes a specific element from a List object.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.remove Method (Int32)

Deletes an element at a specific index from a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object remove(
    int ix);
```

Parameters

ix

The index of the element to be deleted.

Return Value

The deleted element.

Example

In this example, you create and initialize a linked list object, and then you remove the element at index #1.

```
// list-remove1.jsl
// List.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My node");
        lList.add("Her node");
        lList.add("Their node");
        lList.add("Your node");

        // Remove the element at index 1:
        lList.remove(1);

        // Display the elements after the removal:
        System.out.println("The removed element is: " +
                           lList.remove(1) );
    }
}

/*
The removed element is: Their node
*/
```

Remarks

Throws IndexOutOfBoundsException if ix is negative, or greater than or equal to size.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.remove Method (Object)

Deletes a specific element from a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean remove(
    java.lang.Object e);
```

Parameters

e

The element to be deleted.

Return Value

true if the element was successfully deleted; false otherwise.

Example

In this example, you create and initialize a linked list object, and then you remove one element and display the list before and after the removal.

```
// list-remove2.jsl
// List.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My node");
        lList.add("Her node");
        lList.add("Their node");
        lList.add("Your node");

        // Display the elements before the removal:
        System.out.println("The elements are: " + lList);


        // Remove the element at index 2:
        lList.remove("Their node");

        // Display the elements after the removal:
        System.out.println("The elements now are: " + lList);
    }
}

/*
Output:
The elements are: [My node, Her node, Their node, Your node]
The elements now are: [My node, Her node, Your node]
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.removeAll Method

Deletes all the elements from a List object that exist is a specified collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean removeAll(
    java.util.Collection c);
```

Parameters

c

The collection of elements to be deleted from List.

Return Value

true if the list is modified after the call; false otherwise.

Example

In this example, you create and initialize two collections, aList and lList. Then you remove all the elements from lList that exist in aList. The output displays the contents of the two lists before and after the removal.

```
// List-remA1.jsl
// List.removeAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create two collections:
        LinkedList lList =  new LinkedList();
        ArrayList aList =  new ArrayList();

        // Add some elements to lList:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Hazem");

        // Add some elements to aList:
        aList.add("Isabella");
        aList.add("Angelina");
        aList.add("Bianca");

        // Display the two lists (before):
        System.out.println("The linked list before: " + lList);
        System.out.println("The array list before: " + aList);

        // Remove aList members that exist in lList:
        lList.removeAll(aList);

        // Display the two lists (after):
        System.out.println("The linked list after: " + lList);
        System.out.println("The array list after: " + aList);
    }
}

/*
Output:
The linked list before: [Isabella, Angelina, Pille, Hazem]
```

```
The array list before: [Isabella, Angelina, Bianca]
The linked list after: [Pille, Hazem]
The array list after: [Isabella, Angelina, Bianca]
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.retainAll Method

Deletes the elements of the current List collection that do not exist in specified collection object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean retainAll(
    java.util.Collection c);
```

Parameters

c

The specified collection.

Return Value

true if the list has changed after the call; false otherwise.

Example

In this example, you create and initialize two collections, aList and lList. Then you delete the elements of lList that do not exist in aList and display both collections before and after.

```
// List-retA1.jsl
// List.retainAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create two collections:
        LinkedList lList =  new LinkedList();
        ArrayList aList =  new ArrayList();

        // Add some elements to lList:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Pille");
        lList.add("Hazem");

        // Add some elements to aList:
        aList.add("Isabella");
        aList.add("Angelina");
        aList.add("Bianca");

        // Display the two collections before:
        System.out.println("The aList collection before: " + aList);
        System.out.println("The lList collection before: " + lList);

        // Delete the elements lList that are not in aList:
        lList.retainAll(aList);

        // Display the collection after:
        // Display the two collections before retaining all:
        System.out.println("The aList collection after: " + aList);
        System.out.println("The lList collection after: " + lList);
    }
}

/*
Output:
```

```
The aList collection before: [Isabella, Angelina, Bianca]
The lList collection before: [Isabella, Angelina, Pille, Hazem]
The aList collection after: [Isabella, Angelina, Bianca]
The lList collection after: [Isabella, Angelina]
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.set Method

Inserts an element at the specified location.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object set(
    int ix,
    java.lang.Object e);
```

Parameters

ix

The index of the element.

e

The List element to be added.

Return Value

The new element in the List object.

Example

```
// list-set1.jsl
// List.set example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My node");
        lList.add("Her node");
        lList.add("Their node");
        lList.add("Your node");

        // Display the elements:
        System.out.println("The old list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + "=" + lList.get(i));

        // Set the element #2:
           System.out.println("\nThe following element has been set: "
                              + lList.set(2,"His node"));

        // Display the elements after changing #2:
        System.out.println("\nThe new list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + "=" + lList.get(i));
    }
}

/*
Output:
The old list:
0=My node
1=Her node
```

```
2=Their node
3=Your node

The following element has been set: Their node

The new list:
0=My node
1=Her node
2=His node
3=Your node
*/
```

Remarks

Throws IndexOutOfBoundsException if ix is negative, or greater than or equal to size.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.size Method

Retrieves the number of elements in a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int size();
```

Return Value

The number of elements in the List object.

Example

See the example under clear.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.subList Method

Creates a sub list from the elements that exist within a specified index range in a List object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.List subList(
    int fromIx,
    int toIx);
```

Parameters

fromIx

The starting index of the range.

toIx

The ending index of the range.

Return Value

A list that contains the elements in the specified range.

Example

```
// list-sublist1.jsl
// List.subList example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My node");
        lList.add("Her node");
        lList.add("Their node");
        lList.add("Your node");

        // Display the list:
        System.out.println("The original list:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + "=" + lList.get(i));

        // Create a sublist:
        List lList1 = lList.subList(2,4);

        // Display the sublist:
        System.out.println("\nThe sublist:");
        for (int i=0; i<lList1.size(); i++)
            System.out.println(i + "=" + lList1.get(i));
    }
}

/*
Output:
The original list:
0=My node
1=Her node
2=Their node
```

```
3=Your node

The sublist:
0=Their node
1=Your node
*/
```

Remarks

Throws IndexOutOfBoundsException if fromIx is negative, or toIx greater than size.

Throws IllegalArgumentException if toIx is less than fromIx.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.toArray Method

Overload List

Name	Description
List.toArray ()	Copies and returns a List elements into an array.
List.toArray (Object[])	Copies and returns all the elements of a List object into a specific array object.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.toArray Method ()

Copies and returns a List elements into an array.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object[] toArray();
```

Return Value

The array object that contains the elements of the List object.

Example

In this example, you create and initialize a linked list object, and then you convert the list to an array and display the elements of the array.

```
// list-toArr1.jsl
// List.toArray example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My node");
        lList.add("Her node");
        lList.add("Their node");
        lList.add("Your node");

        // Create an array from the list:
        Object[] s = lList.toArray();

        // Display the array elements:
        System.out.println("The array elements are:");
        for (int i=0; i<lList.size(); i++)
            System.out.println(i + "=" + s[i]);
    }
}

/*
Output:
The array elements are:
0=My node
1=Her node
2=Their node
3=Your node
*/
```

See Also

Reference

List Interface

Concepts

List Members

java.util Package

List.toArray Method (Object[ ])

Copies and returns all the elements of a List object into a specific array object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object[] toArray(
    java.lang.Object[] arr);
```

Parameters

arr

The array that receives the elements of the List object.

Return Value

The array object that contains the elements of the List object.

Example

In this example, you create and initialize a linked list object, and then you convert the list to an array "s". Then you copy "s" to a new array "s1" and display the elements of "s1."

```
// list-toArr2.jsl
// List.toArray example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("My node");
        lList.add("Her node");
        lList.add("Their node");
        lList.add("Your node");

        // Create an array from the list:
        String[] arr = new String[0];

        // Copy lList to arr:
        arr = (String[])lList.toArray(arr);

        // Display the new array elements:
        System.out.println("The array elements are:");
        for (int i=0; i<arr.length; i++)
            System.out.println(i + "=" + arr[i]);
    }
}

/*
Output:
The array elements are:
0=My node
1=Her node
2=Their node
3=Your node
*/
```

Remarks

If arr is not large enough to hold the List members it will be expanded.

The type of List elements should match the type of the array elements.

See Also

Reference

List Interface

Concepts

List Members

java.util Package

ListIterator Interface

An interface that implements the Iterator interface. It allows traversing the list and modifying it during iteration. It also obtains the current position of the iterator's cursor. The cursor position is between the elements of a collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.ListIterator
    extends java.util.Iterator
```

Remarks

This interface is not thread safe. If two threads modify the list concurrently, the user will get ConcurrentModificationException.

See Also

Concepts

ListIterator Members

java.util Package

ListIterator Members

An interface that implements the Iterator interface. It allows traversing the list and modifying it during iteration. It also obtains the current position of the iterator's cursor. The cursor position is between the elements of a collection.

The following tables list the members exposed by the ListIterator type.

Public Methods

Name	Description
add	Adds an object to ListIterator.
hasNext	Checks if there exists an element beyond the iterator position in a ListIterator object.
hasPrevious	Checks if there exists an element before the iterator position in a ListIterator object.
next	Returns the next object beyond the iterator position in a ListIterator object and moves the iterator's position.
nextIndex	Returns the index of the next object in a ListIterator object.
previous	Returns the object before the iterator position in a ListIterator object and moves the iterator's position.
previousIndex	Returns the index of the previous object in a ListIterator object.
remove	Removes the last element from a ListIterator object, which was called by next or previous.
set	Sets an element in a ListIterator collection.

See Also

Reference

ListIterator Interface

Concepts

java.util Package

ListIterator Methods

Public Methods

Name	Description
add	Adds an object to ListIterator.
hasNext	Checks if there exists an element beyond the iterator position in a ListIterator object.
hasPrevious	Checks if there exists an element before the iterator position in a ListIterator object.
next	Returns the next object beyond the iterator position in a ListIterator object and moves the iterator's position.
nextIndex	Returns the index of the next object in a ListIterator object.
previous	Returns the object before the iterator position in a ListIterator object and moves the iterator's position.
previousIndex	Returns the index of the previous object in a ListIterator object.
remove	Removes the last element from a ListIterator object, which was called by next or previous.
set	Sets an element in a ListIterator collection.

See Also

Reference

ListIterator Interface

Concepts

java.util Package

ListIterator.add Method

Adds an object to ListIterator.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void add(
    java.lang.Object e);
```

Parameters

e

The object to be added to ListIterator.

Example

In this example, you create a ListIterator object before the second element in a collection, and add a new element at this position.

```
// LList-Add1.jsl
// ListIteratort.add example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList = new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Get a list iterator before index 1:
        ListIterator li = lList.listIterator(1);

        // Add a new element at index 1:
        li.add("Bianca");

        // Display the collection:
        System.out.println("The collection is: " + lList);
    }
}

/*
Output:
The collection is: [Isabella, Bianca, Angelina]
*/
```

See Also

Reference

ListIterator Interface

Concepts

ListIterator Members

java.util Package

ListIterator.hasNext Method

Checks if there exists an element beyond the iterator position in a ListIterator object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean hasNext();
```

Return Value

true if there is a next element; false otherwise.

Example

In this example, you create and initialize a ListIterator object before the second element in a collection, and then you display all elements next to the iterator's position.

```
// LIter-hnext1.jsl
// ListIteratort.hasNext example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList = new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Bianca");

        // Get a list iterator before index 1:
        ListIterator li = lList.listIterator(1);

        // Display the next elements:
        while(li.hasNext())
            System.out.println(li.next());
    }
}

/*
Output:
Angelina
Bianca
*/
```

See Also

Reference

ListIterator Interface

Concepts

ListIterator Members

java.util Package

ListIterator.hasPrevious Method

Checks if there exists an element before the iterator position in a ListIterator object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean hasPrevious();
```

Return Value

true if there is a previous element; false otherwise.

Example

In this example, you create a ListIterator object before the second element in a collection, and then you check to see if there is a previous element.

```
// LIter-hPrev1.jsl
// ListIteratort.hasPrevious example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList = new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Get a list iterator before index 1:
        ListIterator li = lList.listIterator(1);

        // Display if there is a previous element:
        System.out.println(li.hasPrevious());
    }
}

/*
Output:
true
*/
```

See Also

Reference

ListIterator Interface

Concepts

ListIterator Members

java.util Package

ListIterator.next Method

Returns the next object beyond the iterator position in a ListIterator object and moves the iterator's position.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object next();
```

Return Value

The next object beyond the iterator position in ListIterator.

Example

In this example, you create a ListIterator object before the second element in a collection, and then you display the element next to the iterator's position.

```
// LIter-next1.jsl
// ListIteratort.next example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList = new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Get a list iterator before index 1:
        ListIterator li = lList.listIterator(1);

        // Display the next element:
        System.out.println("The next element is: " + li.next());
    }
}

/*
Output:
The next element is: Angelina
*/
```

Remarks

Throws NoSuchElementException if there is no next object or if the list is null.

See Also

Reference

ListIterator Interface

Concepts

ListIterator Members

java.util Package

ListIterator.nextIndex Method

Returns the index of the next object in a ListIterator object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int nextIndex();
```

Return Value

The index of the next object in the ListIterator object.

Example

In this example, you create a ListIterator object before the second element in a collection, and then you display the index of the element next to the iterator's position.

```
// LIter-nxtix1.jsl
// ListIteratort. nextIndex example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList = new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Get a list iterator before index 1:
        ListIterator li = lList.listIterator(1);

        // Display the next index:
        System.out.println("Next index: " + li.nextIndex());
    }
}

/*
Output:
Next index: 1
*/
```

See Also

Reference

ListIterator Interface

Concepts

ListIterator Members

java.util Package

ListIterator.previous Method

Returns the object before the iterator position in a ListIterator object and moves the iterator's position.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object previous();
```

Return Value

The object before the iterator position in the ListIterator object.

Example

In this example, you create a ListIterator object before the second element in a collection, and then you display the element before the iterator's position.

```
// LIter-prev1.jsl
// ListIteratort.previous example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList = new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Get a list iterator:
        ListIterator li = lList.listIterator(1);

        // Display the previous element:
        System.out.println("The previous element is: " +
                           li.previous());
    }
}

/*
Output:
The previous element is: Isabella
*/
```

Remarks

Throws NoSuchElementException if there is no previous object or if the list is null.

See Also

Reference

ListIterator Interface

Concepts

ListIterator Members

java.util Package

ListIterator.previousIndex Method

Returns the index of the previous object in a ListIterator object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int previousIndex();
```

Return Value

The index of the previous object in the ListIterator object.

Example

In this example, you create a ListIterator object before the second element in a collection, and then you display the index of the element before the iterator's position.

```
// LIter-previx1.jsl
// ListIteratort.previousIndex example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList = new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");

        // Get a list iterator:
        ListIterator li = lList.listIterator(1);

        // Display the previous index:
        System.out.println("Previous index: " + li.previousIndex());
    }
}

/*
Output:
Previous index: 0
*/
```

See Also

Reference

ListIterator Interface

Concepts

ListIterator Members

java.util Package

ListIterator.remove Method

Removes the last element from a ListIterator object, which was called by next or previous.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void remove();
```

Example

In this example, you construct a collection and create a ListIterator before position 0. A call to next, at this point, will return the first element in the collection. Then you remove the first element using remove, and display the new collection and the new next element.

```
// LIter-rem1.jsl
// ListIterator.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList = new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Bianca");

        // Get a list iterator before the starting index:
        ListIterator li = lList.listIterator();

        // Display the list:
        System.out.println("The collection: " + lList);

        // Dispaly the next element:
        System.out.println("The next element: " + li.next());

         // Remove the last called by next():
        li.remove();

        // Display the new list:
        System.out.println("The new collection: " + lList);

        // Display the new next element:
        System.out.println("The new next element: " + li.next());
    }
}

/*
Output:
The collection: [Isabella, Angelina, Bianca]
The next element: Isabella
The new collection: [Angelina, Bianca]
The new next element: Angelina
*/
```

Remarks

Throws IllegalStateException if no call to next or previous has been made.

See Also

Reference

ListIterator Interface

Concepts

ListIterator Members

java.util Package

ListIterator.set Method

Sets an element in a ListIterator collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void set(
    java.lang.Object e);
```

Parameters

e

The object to be set.

Example

In this example, you construct a collection and create a ListIterator before position 0. A call to next, at this point, will return the first element in the collection. Then you change the first element by using set, and display the new collection.

```
// LIter-set1.jsl
// ListIterator.set example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList = new LinkedList();

        // Add some elements:
        lList.add("Isabella");
        lList.add("Angelina");
        lList.add("Bianca");

        // Get a list iterator before the starting index:
        ListIterator li = lList.listIterator();

        // Display the list:
        System.out.println("The collection: " + lList);

        // Dispaly the next element:
        System.out.println("The next element: " + li.next());

         // Change the last called by next():
        li.set("Sally");

        // Display the new list:
        System.out.println("The new collection: " + lList);
    }
}

/*
Output:
The collection: [Isabella, Angelina, Bianca]
The next element: Isabella
The new collection: [Sally, Angelina, Bianca]
*/
```

Remarks

Throws IllegalStateException if no call to next or previous has been made.

See Also

Reference

ListIterator Interface

Concepts

ListIterator Members

java.util Package

ListResourceBundle Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.util.ListResourceBundle
    extends java.util.ResourceBundle
```

Inheritance Hierarchy

java.lang.Object

  java.util.ResourceBundle

   java.util.ListResourceBundle

See Also

Concepts

ListResourceBundle Members

java.util Package

ListResourceBundle Members

The following tables list the members exposed by the ListResourceBundle type.

Public Constructors

Name	Description
ListResourceBundle	

Public Fields

Name	Description
parent	(inherited from **ResourceBundle**)

Public Methods

Name	Description
equals	(inherited from Object)
getContents	
hashCode	(inherited from Object)
getKeys	Overridden.
getObject	(inherited from **ResourceBundle**)
getString	(inherited from **ResourceBundle**)
getStringArray	(inherited from **ResourceBundle**)
getClass	(inherited from Object)
handleGetObject	Overridden.
clone	(inherited from **ResourceBundle**)
setParent	(inherited from **ResourceBundle**)
toString	(inherited from **ResourceBundle**)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ListResourceBundle Class

Concepts

java.util Package

ListResourceBundle Fields

Public Fields

Name	Description
parent	(inherited from **java.util.ResourceBundle**)

See Also

Reference

ListResourceBundle Class

Concepts

java.util Package

ListResourceBundle Constructor

Initializes a new instance of the ListResourceBundle Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.ListResourceBundle();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

ListResourceBundle Class

Concepts

ListResourceBundle Members

java.util Package

ListResourceBundle Methods

Public Methods

Name	Description
equals	(inherited from Object)
getContents	
hashCode	(inherited from Object)
getKeys	Overridden.
getObject	(inherited from **ResourceBundle**)
getString	(inherited from ResourceBundle)
getStringArray	(inherited from **ResourceBundle**)
getClass	(inherited from Object)
handleGetObject	Overridden.
clone	(inherited from **ResourceBundle**)
setParent	(inherited from **ResourceBundle**)
toString	(inherited from **ResourceBundle**)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

ListResourceBundle Class

Concepts

java.util Package

ListResourceBundle.getContents Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected abstract [Ljava.lang.Object;[] getContents();
```

See Also

Reference

ListResourceBundle Class

Concepts

ListResourceBundle Members

java.util Package

ListResourceBundle.getKeys Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Enumeration getKeys();
```

See Also

Reference

ListResourceBundle Class

Concepts

ListResourceBundle Members

java.util Package

ListResourceBundle.handleGetObject Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.Object handleGetObject(
    java.lang.String key);
```

Parameters

key

See Also

Reference

ListResourceBundle Class

Concepts

ListResourceBundle Members

java.util Package

Locale Class (J#)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final class java.util.Locale
    extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable, System.Runtime.Serialization.ISer
ializable
```

Inheritance Hierarchy

java.lang.Object

java.util.Locale

See Also

Concepts

Locale Members

java.util Package

Locale Members (J#)

The following tables list the members exposed by the Locale type.

Public Constructors

Name	Description
Locale	Overloaded.

Public Fields

Name	Description
CANADA	
CANADA_FRENCH	
CHINA	
CHINESE	
ENGLISH	
FRANCE	
FRENCH	
GERMAN	
GERMANY	
ITALIAN	
ITALY	
JAPAN	
JAPANESE	
KOREA	
KOREAN	
PRC	
SIMPLIFIED_CHINESE	
TAIWAN	
TRADITIONAL_CHINESE	
UK	
US	

Public Methods

Name	Description
equals	Overridden.
getCountry	
getDefault	
getDisplayCountry	Overloaded.
getDisplayLanguage	Overloaded.
getDisplayName	Overloaded.
getDisplayVariant	Overloaded.
hashCode	Overridden.
getISO3Country	
getISO3Language	
getLanguage	
GetObjectData	
getClass	(inherited from Object)
getVariant	
clone	
setDefault	
toString	Overridden.

See Also

Reference

Locale Class

Concepts

java.util Package

Locale Fields

Public Fields

Name	Description
CANADA	
CANADA_FRENCH	
CHINA	
CHINESE	
ENGLISH	
FRANCE	
FRENCH	
GERMAN	
GERMANY	
ITALIAN	
ITALY	
JAPAN	
JAPANESE	
KOREA	
KOREAN	
PRC	
SIMPLIFIED_CHINESE	
TAIWAN	
TRADITIONAL_CHINESE	
UK	
US	

See Also

Reference

Locale Class

Concepts

java.util Package

Locale.CANADA Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale CANADA;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.CANADA_FRENCH Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale CANADA_FRENCH;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.CHINA Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale CHINA;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.CHINESE Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale CHINESE;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.ENGLISH Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale ENGLISH;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.FRANCE Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale FRANCE;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.FRENCH Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale FRENCH;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.GERMAN Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale GERMAN;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.GERMANY Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale GERMANY;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.ITALIAN Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale ITALIAN;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.ITALY Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale ITALY;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.JAPAN Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale JAPAN;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.JAPANESE Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale JAPANESE;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.KOREA Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale KOREA;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.KOREAN Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale KOREAN;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.PRC Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale PRC;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.SIMPLIFIED_CHINESE Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale SIMPLIFIED_CHINESE;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.TAIWAN Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale TAIWAN;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.TRADITIONAL_CHINESE Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale TRADITIONAL_CHINESE;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.UK Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale UK;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.US Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static final java.util.Locale US;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale Constructor

Overload List

Name	Description
Locale (SerializationInfo, StreamingContext)	
Locale (String, String)	
Locale (String, String, String)	

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale Constructor (SerializationInfo, StreamingContext)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.Locale(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also
Reference
Locale Class
Concepts
Locale Members
java.util Package

Locale Constructor (String, String)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Locale(
    java.lang.String languageCode,
    java.lang.String ctryRgnCode);
```

Parameters

languageCode

ctryRgnCode

See Also
Reference
Locale Class
Concepts
Locale Members
java.util Package

Locale Constructor (String, String, String)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Locale(
    java.lang.String languageCode,
    java.lang.String ctryRgnCode,
    java.lang.String variantCode);
```

Parameters

languageCode

ctryRgnCode

variantCode

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale Methods

Public Methods

Name	Description
equals	Overridden.
getCountry	
getDefault	
getDisplayCountry	Overloaded.
getDisplayLanguage	Overloaded.
getDisplayName	Overloaded.
getDisplayVariant	Overloaded.
hashCode	Overridden.
getISO3Country	
getISO3Language	
getLanguage	
GetObjectData	
getClass	(inherited from Object)
getVariant	
clone	
setDefault	
toString	Overridden.

See Also

Reference

Locale Class

Concepts

java.util Package

Locale.clone Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.equals Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getCountry Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getCountry();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDefault Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static java.util.Locale getDefault();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayCountry Method

Overload List

Name	Description
Locale.getDisplayCountry ()	
Locale.getDisplayCountry (Locale)	

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayCountry Method ()

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String getDisplayCountry();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayCountry Method (Locale)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getDisplayCountry(
    java.util.Locale inLocale);
```

Parameters

inLocale

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayLanguage Method

Overload List

Name	Description
Locale.getDisplayLanguage ()	
Locale.getDisplayLanguage (Locale)	

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayLanguage Method ()

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String getDisplayLanguage();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayLanguage Method (Locale)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getDisplayLanguage(
    java.util.Locale inLocale);
```

Parameters

inLocale

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayName Method

Overload List

Name	Description
Locale.getDisplayName ()	
Locale.getDisplayName (Locale)	

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayName Method ()

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String getDisplayName();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayName Method (Locale)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getDisplayName(
    java.util.Locale inLocale);
```

Parameters

inLocale

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayVariant Method

Overload List

Name	Description
Locale.getDisplayVariant ()	
Locale.getDisplayVariant (Locale)	

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayVariant Method ()

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String getDisplayVariant();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getDisplayVariant Method (Locale)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getDisplayVariant(
    java.util.Locale inLocale);
```

Parameters

inLocale

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.hashCode Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized int hashCode();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getISO3Country Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getISO3Country() throws java.util.MissingResourceException;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getISO3Language Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getISO3Language() throws java.util.MissingResourceException;
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getLanguage Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getLanguage();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.GetObjectData Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.getVariant Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getVariant();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.setDefault Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized void setDefault(
    java.util.Locale inLocale);
```

Parameters

inLocale

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Locale.toString Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.String toString();
```

See Also

Reference

Locale Class

Concepts

Locale Members

java.util Package

Map Interface

An interface that provides a Map object for key entries and their corresponding values.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.Map
```

See Also

Concepts

Map Members

java.util Package

Map Members (J#)

An interface that provides a Map object for key entries and their corresponding values.

The following tables list the members exposed by the Map type.

Public Methods

Name	Description
clear	Removes the elements from a Map object.
containsKey	Checks for a specific key in a Map object.
containsValue	Checks for a specific value in a Map object.
entrySet	Retrieves the set of mappings contained in an entrySet object.
equals	Checks if a specified element in a Map object is equal to a specified object.
get	Retrieves an element from a Map object.
hashCode	Retrieves the hash code for a Map object.
isEmpty	Checks if a Map object is empty.
keySet	Retrieves the key values of the keys in a Map object.
put	Adds an element at a specific key entry to a Map object.
putAll	Copies all the elements from a specified map to the current Map object.
remove	Deletes an element associated with a specific key from a Map object.
size	Retrieves the number of the key-value mappings of a Map object.
values	Retrieves the collection of values in a Map object.

See Also

Reference

Map Interface

Concepts

java.util Package

Map Methods

Public Methods

Name	Description
clear	Removes the elements from a Map object.
containsKey	Checks for a specific key in a Map object.
containsValue	Checks for a specific value in a Map object.
entrySet	Retrieves the set of mappings contained in an entrySet object.
equals	Checks if a specified element in a Map object is equal to a specified object.
get	Retrieves an element from a Map object.
hashCode	Retrieves the hash code for a Map object.
isEmpty	Checks if a Map object is empty.
keySet	Retrieves the key values of the keys in a Map object.
put	Adds an element at a specific key entry to a Map object.
putAll	Copies all the elements from a specified map to the current Map object.
remove	Deletes an element associated with a specific key from a Map object.
size	Retrieves the number of the key-value mappings of a Map object.
values	Retrieves the collection of values in a Map object.

See Also

Reference

Map Interface

Concepts

java.util Package

Map.clear Method

Removes the elements from a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void clear();
```

Example

In the following example, you create and initialize a HashMap object, which implements Map, and then you remove all the elements and display the values. The result is an empty set.

```
// Map-clear1.jsl
// Map.clear example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashMap object (Implements Map):
        HashMap hm = new HashMap();

        // Add some elements:
        hm.put("4","Eve Kennedy");
        hm.put("1","Isabella Abolrous");
        hm.put("2","Emma Esteban");

        // Clear the elements:
        hm.clear();

        // Display the values:
        System.out.println("The values are: " + hm.values());
    }
}

/*
The values are: []
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.containsKey Method

Checks for a specific key in a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean containsKey(
    java.lang.Object k);
```

Parameters

k

The key to search for in the Map object.

Return Value

true if the key exists; false otherwise.

Example

In the following example, you create and initialize a HashMap object, which implements Map, and then you check for the key "3" The result is false because this key is not in the object.

```
// Map-conkey1.jsl
// Map.containsKey example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashMap object (Implements Map):
        HashMap hm = new HashMap();

        // Add some elements:
        hm.put("4","Eve Kennedy");
        hm.put("1","Isabella Abolrous");
        hm.put("2","Emma Esteban");

        // Check for a specific key:
        System.out.println(hm.containsKey("3"));
    }
}

/*
false
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.containsValue Method

Checks for a specific value in a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean containsValue(
    java.lang.Object v);
```

Parameters

v

The value to search for in the Map object.

Return Value

true if the value exists; false otherwise.

Example

In the following example, you create and initialize a HashMap object, which implements Map, and then you check for the value "Eve Kennedy." The result is true.

```
// Map-conValue1.jsl
// Map.containsValue example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashMap object (Implements Map):
        HashMap hm = new HashMap();

        // Add some elements:
        hm.put("4","Eve Kennedy");
        hm.put("1","Isabella Abolrous");
        hm.put("2","Emma Esteban");

        // Check for a specific value:
        System.out.println(hm.containsValue("Eve Kennedy"));
    }
}

/*
true
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.entrySet Method

Retrieves the set of mappings contained in an entrySet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Set entrySet();
```

Return Value

The set of mappings in the entrySet object.

Example

In this example, you create and initialize a TreeMap object, which implements Map, and then you display the set of mappings in the object.

```
// Map-entrysm1.jsl
// Map.entrySet example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object (implements Map):
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("3","Craig Combel");
        tm.put("5","Pille Mandla");

        // Display the entry set of the object:
        System.out.println("The entry set is:\n" + tm.entrySet());
    }
}

/*
Output:
The entry set is:
[[1, Sally Abolrous], [3, Craig Combel], [5, Pille Mandla]]
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.equals Method

Checks if a specified element in a Map object is equal to a specified object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean equals(
    java.lang.Object c);
```

Parameters

c

The element to compare to.

Return Value

true if the compared elements are the same; false otherwise.

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.get Method

Retrieves an element from a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object get(
    java.lang.Object k);
```

Parameters

k

The key of the element to be retrieved from the Map object.

Return Value

The object to be retrieved from the Map object.

Example

In this example, you create and initialize a TreeMap object, which implements Map, and then you display the element associated with the key "2."

Code

```
// Map-get1.jsl
// Map.get example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object (implements Map):
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1", "Sally Abolrous");
        tm.put("2","Craig Combel");

        // Display the element that corresponds to the key #2:
        System.out.println("The element #2 is: " + tm.get("2"));
    }
}

/*
Output:
The element #2 is: Craig Combel
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.hashCode Method

Retrieves the hash code for a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int hashCode();
```

Return Value

An integer that represents the hash code of the Map object.

Example

In the following example, you create and initialize a Hashtable object, which implements Map, and then you display the hash code of the object.

```
// Map-hashcode1.jsl
// Map.GetHashCode example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a Hashtable object (implements Map):
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("1","Eve Kennedy");
        ht.put("2","Audrey Esteban");
        ht.put("5","Emma Esteban");

        // Display the hash code:
        System.out.println("The hash code is: " + ht.GetHashCode());
    }
}

/*
The hash code is: 1334170275
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.isEmpty Method

Checks if a Map object is empty.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isEmpty();
```

Return Value

true if the Map object is empty; false otherwise.

Example

In this example, you create and initialize a TreeMap object, and then you check if the object is empty. You clear the object and check again. The test in the latter case returns true.

```
// Map-isEmpty1.jsl
// Map.isEmpty example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("3","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("5","Pille Mandla");

        // Check if the object is empty:
        System.out.println(tm.isEmpty());

        // Clear the object:
        tm.clear();

        // Check if the object is empty:
        System.out.println(tm.isEmpty());
    }
}

/*
Output:
false
true
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.keySet Method

Retrieves the key values of the keys in a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Set keySet();
```

Return Value

The set of the key entries in the Map object.

Example

In the following example, you create and initialize a Hashtable object, which implements Map, and then you display the keys stored in the object.

```
// Map-keyset1.jsl
// Map.keySet example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a Hashtable object (implements Map):
        Hashtable ht = new Hashtable();

        // Add some elements:
        ht.put("1","Eve Kennedy");
        ht.put("2","Audrey Esteban");
        ht.put("5","Emma Esteban");

        // Display the keys:
        System.out.println("The keys are: " + ht.keySet());
    }
}

/*
The keys are: [1, 2, 5]
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.put Method

Adds an element at a specific key entry to a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object put(
    java.lang.Object k,
    java.lang.Object v);
```

Parameters

k

The key entry at which the element is added.

v

The element to be added.

Return Value

The element to be added to Map.

Example

In the following example, you create and initialize a HashMap object, which implements Map, and then you display the values stored in the object.

```
// Map-put1.jsl
// Map.put example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashMap object (Implements Map):
        HashMap hm = new HashMap();

        // Add some elements:
        hm.put("4","Eve Kennedy");
        hm.put("1","Isabella Abolrous");
        hm.put("2","Emma Esteban");

        // Display the values:
        System.out.println("The values are: " + hm.values());
    }
}

/*
The values are: [Isabella Abolrous, Emma Esteban, Eve Kennedy]
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.putAll Method

Copies all the elements from a specified map to the current Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void putAll(
    java.util.Map m);
```

Parameters

m

The specified Map.

Example

In this example, you create and initialize a TreeMap object, and then you copy it to a HashMap object. Both objects implement the Map interface.

```
// Map-putAll1.jsl
// Map.putAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();
        // Create a HashMap object:
        HashMap ht = new HashMap();

        // Add some elements to tm:
        tm.put("1","Sally Abolrous");
        tm.put("2","Craig Combel");

        // Copy tm to ht:
        ht.putAll(tm);

        // Display the element #2 in both objects:
        System.out.println("The element #2 in tm is: "
                           + tm.get("2"));
        System.out.println("The element #2 in ht is: "
                           + ht.get("2"));
    }
}

/*
Output:
The element #2 in tm is: Craig Combel
The element #2 in ht is: Craig Combel
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.remove Method

Deletes an element associated with a specific key from a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object remove(
    java.lang.Object k);
```

Parameters

k

The key associated with the element to be removed from the Map object.

Return Value

The removed object.

Example

Description

In this example, you create and initialize a TreeMap object, which implements Map, and then you remove one element from the object.

Code

```
// Map-rem1.jsl
// Map.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object (implements Map):
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("3","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("5","Pille Mandla");

        // Remove an object:
        System.out.println("The following object has been removed: " +
                           tm.remove("5"));
    }
}

/*
Output:
The following object has been removed: Pille Mandla
```

*/

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.size Method

Retrieves the number of the key-value mappings of a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int size();
```

Return Value

An integer that represents the number of key-value mappings of the Map object.

Example

In this example, you create and initialize a TreeMap object, which implements Map, and then you display the number of the key-value mappings of the object.

Code

```
// Map-size1.jsl
// Map.size example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("3","Pille Mandla");

        // Display the size of the TreeMap:
        System.out.println("The size is: " + tm.size());
    }
}

/*
Output:
The size is: 3
*/
```

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.values Method

Retrieves the collection of values in a Map object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Collection values();
```

Return Value

The collection of values in the Map object.

Example

See the example on put.

See Also

Reference

Map Interface

Concepts

Map Members

java.util Package

Map.Entry Interface

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.Map.Entry
```

See Also

Concepts

Map.Entry Members

java.util Package

Map.Entry Members

The following tables list the members exposed by the Map.Entry type.

Public Methods

Name	Description
equals	
hashCode	
getKey	
getValue	
setValue	

See Also

Reference

Map.Entry Interface

Concepts

java.util Package

Map.Entry Methods

Public Methods

Name	Description
equals	
hashCode	
getKey	
getValue	
setValue	

See Also

Reference

Map.Entry Interface

Concepts

java.util Package

Map.Entry.equals Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean equals(
    java.lang.Object e);
```

Parameters

e

See Also

Reference

Map.Entry Interface

Concepts

Map.Entry Members

java.util Package

Map.Entry.hashCode Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int hashCode();
```

See Also

Reference

Map.Entry Interface

Concepts

Map.Entry Members

java.util Package

Map.Entry.getKey Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object getKey();
```

See Also

Reference

Map.Entry Interface

Concepts

Map.Entry Members

java.util Package

Map.Entry.getValue Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object getValue();
```

See Also

Reference

Map.Entry Interface

Concepts

Map.Entry Members

java.util Package

Map.Entry.setValue Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object setValue(
    java.lang.Object newVal);
```

Parameters

newVal

See Also

Reference

Map.Entry Interface

Concepts

Map.Entry Members

java.util Package

MissingResourceException Class

An exception thrown to indicate that a resource is missing.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.MissingResourceException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.util.MissingResourceException

See Also

Concepts

MissingResourceException Members

java.util Package

MissingResourceException Members

An exception thrown to indicate that a resource is missing.

The following tables list the members exposed by the MissingResourceException type.

Public Constructors

Name	Description
MissingResourceException	Overloaded. Constructs a MissingResourceException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
getClassName	Retrieves the class name of the missing resource that led to MissingResourceException.
hashCode	(inherited from Object)
getKey	Retrieves the key of the missing resource that led to MissingResourceException.
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden. Retrieves the data required to serialize a MissingResourceException object.
getClass	(inherited from Object)

clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

MissingResourceException Class

Concepts

java.util Package

MissingResourceException Constructor

Constructs a MissingResourceException object.

Overload List

Name	Description
MissingResourceException (SerializationInfo, StreamingContext)	Constructs a MissingResourceException object during serialization.
MissingResourceException (String, Exception)	Constructs a MissingResourceException object with a specific text message and inner exception.
MissingResourceException (String, String, String)	Constructs a MissingResourceException object with specific information about the missing resource.
MissingResourceException (String, String, String, Exception)	Constructs a MissingResourceException object with specific information about the missing resource..

See Also

Reference

MissingResourceException Class

Concepts

MissingResourceException Members

java.util Package

MissingResourceException Constructor (SerializationInfo, StreamingContext)

Constructs a MissingResourceException object during serialization.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.MissingResourceException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing an object.

context

The source or destination for the serialization.

See Also

Reference

MissingResourceException Class

Concepts

MissingResourceException Members

java.util Package

MissingResourceException Constructor (String, Exception)

Constructs a MissingResourceException object with a specific text message and inner exception.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.MissingResourceException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

The string that contains the message text.

inner

The inner exception that led to MissingResourceException.

See Also

Reference

MissingResourceException Class

Concepts

MissingResourceException Members

java.util Package

MissingResourceException Constructor (String, String, String)

Constructs a MissingResourceException object with specific information about the missing resource.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.MissingResourceException(
    java.lang.String msg,
    java.lang.String className,
    java.lang.String key);
```

Parameters

msg

The string that contains the message text.

className

The name of the resource class.

key

The key of the missing resource.

See Also

Reference

MissingResourceException Class

Concepts

MissingResourceException Members

java.util Package

MissingResourceException Constructor (String, String, String, Exception)

Constructs a MissingResourceException object with specific information about the missing resource..

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.MissingResourceException(
    java.lang.String msg,
    java.lang.String className,
    java.lang.String key,
    System.Exception inner);
```

Parameters

msg

The string that contains the message text.

className

The name of the resource class.

key

The key of the missing resource.

inner

The inner exception that led to MissingResourceException.

See Also

Reference

MissingResourceException Class

Concepts

MissingResourceException Members

java.util Package

MissingResourceException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
getClassName	Retrieves the class name of the missing resource that led to MissingResourceException.
hashCode	(inherited from Object)
getKey	Retrieves the key of the missing resource that led to MissingResourceException.
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	Overridden. Retrieves the data required to serialize a MissingResourceException object.
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

MissingResourceException Class

Concepts

java.util Package

MissingResourceException.getClassName Method

Retrieves the class name of the missing resource that led to MissingResourceException.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getClassName();
```

Return Value

The class name of the missing resource.

See Also

Reference

MissingResourceException Class

Concepts

MissingResourceException Members

java.util Package

MissingResourceException.getKey Method

Retrieves the key of the missing resource that led to MissingResourceException.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getKey();
```

Return Value

The key string of the missing resource.

See Also

Reference

MissingResourceException Class

Concepts

MissingResourceException Members

java.util Package

MissingResourceException.GetObjectData Method

Retrieves the data required to serialize a MissingResourceException object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information required for serializing the object.

context

The source or destination for the serialization.

See Also

Reference

MissingResourceException Class

Concepts

MissingResourceException Members

java.util Package

MissingResourceException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

MissingResourceException Class

Concepts

java.util Package

NoSuchElementException Class

The exception that is thrown when attempting to access an element in a collection that does not exist. This exception is frequently thrown when attempting to perform an operation on an empty collection or when attempting to access past the end of the collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.NoSuchElementException
    extends java.lang.RuntimeException
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.lang.RuntimeException

java.util.NoSuchElementException

See Also

Concepts

NoSuchElementException Members

java.util Package

NoSuchElementException Members

The exception that is thrown when attempting to access an element in a collection that does not exist. This exception is frequently thrown when attempting to perform an operation on an empty collection or when attempting to access past the end of the collection.

The following tables list the members exposed by the NoSuchElementException type.

Public Constructors

Name	Description
NoSuchElementException	Overloaded. Initializes a new instance of a NoSuchElementException object.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)

toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchElementException Class

Concepts

java.util Package

NoSuchElementException Constructor

Initializes a new instance of a NoSuchElementException object.

Overload List

Name	Description
NoSuchElementException ()	Initializes a new instance of a NoSuchElementException object.
NoSuchElementException (String)	Initializes a new instance of a NoSuchElementException object with the given message.
NoSuchElementException (SerializationInfo, StreamingContext)	Initializes a new instance of a NoSuchElementException object during deserialization.
NoSuchElementException (String, Exception)	Initializes a new instance of a NoSuchElementException object with the given message and inner exception.

See Also

Reference

NoSuchElementException Class

Concepts

NoSuchElementException Members

java.util Package

NoSuchElementException Constructor ()

Initializes a new instance of a NoSuchElementException object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.NoSuchElementException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

NoSuchElementException Class

Concepts

NoSuchElementException Members

java.util Package

NoSuchElementException Constructor (String)

Initializes a new instance of a NoSuchElementException object with the given message.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.NoSuchElementException(
    java.lang.String s);
```

Parameters

s

A message describing the exceptional condition.

See Also

Reference

NoSuchElementException Class

Concepts

NoSuchElementException Members

java.util Package

NoSuchElementException Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a NoSuchElementException object during deserialization.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.NoSuchElementException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

NoSuchElementException Class

Concepts

NoSuchElementException Members

java.util Package

NoSuchElementException Constructor (String, Exception)

Initializes a new instance of a NoSuchElementException object with the given message and inner exception.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.NoSuchElementException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

A message describing the exceptional condition.

inner

An inner exception that led to the NoSuchElementException object being thrown.

See Also

Reference

NoSuchElementException Class

Concepts

NoSuchElementException Members

java.util Package

NoSuchElementException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

NoSuchElementException Class

Concepts

java.util Package

NoSuchElementException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

NoSuchElementException Class

Concepts

java.util Package

Observable Class

Represents the observable object, which usually contains data. It can have one or more observers. An Observer object is an instance of a class that implements the Observer interface. When the data on the observable object changes, the observers can be notified through the update method, which passes the necessary information to the notifyObservers method.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.Observable
    extends java.lang.Object
```

Example

The following example demonstrates the use of most of the methods on the Observable and Observer class. It creates a linked list on the observable object and the changes are sent to the observer object.

```
// Observer-Observable general example

import java.util.*;

class MyClass implements Observer
{
    public void update(Observable o, Object arg)
    {
        Object ob = (LinkedList)arg;
        System.out.println("The list contains: " + ob);
    }

    public static void main(String[] args)
    {
        MyClass mc = new MyClass();
        YourClass yc = new YourClass();
        yc.addObserver(mc);
        System.out.println("Number of observers: " +
                           yc.countObservers());

        // Start watching:
        yc.Watch();
    }
}

class YourClass extends Observable
{
    void Watch()
    {
        // Declare a linked list:
        LinkedList lList = new LinkedList();
        lList.add("Sam");

        // Pass the info the observer:
        setChanged();
        notifyObservers(lList);

        // Check for a change:
        System.out.println("Observable changed?: " + hasChanged());

        // Add nother element:
        lList.add("Pam");

        // Send the new linked list information:
        setChanged();
        notifyObservers(lList);
```

```
        // Set the new change:
        setChanged();


        // Check for a change:
        System.out.println("Observable changed?: " + hasChanged());
    }
}

/*
Output:
Number of observers: 1
The list contains: [Sam]
Observable changed?: false
The list contains: [Sam, Pam]
Observable changed?: true
*/
```

Inheritance Hierarchy

java.lang.Object

java.util.Observable

See Also

Concepts

Observable Members

java.util Package

Observable Members

Represents the observable object, which usually contains data. It can have one or more observers. An Observer object is an instance of a class that implements the Observer interface. When the data on the observable object changes, the observers can be notified through the update method, which passes the necessary information to the notifyObservers method.

The following tables list the members exposed by the Observable type.

Public Constructors

Name	Description
Observable	Constructs a new Observable object.

Public Methods

Name	Description
addObserver	Adds an observer to the Observable object, if it is not already in the list.
clearChanged	Causes the method hasChanged to return false, either because the current Observable object has not change d anymore or because the observer has been already notified of the last change.
countObservers	Returns the number of observers for the current Observable object.
deleteObserver	Deletes a specific observer from the list of observers for the current Observable object.
deleteObservers	Deletes all the observers for the current Observable object.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
hasChanged	Checks if the current Observable object has changed.
clone	
notifyObservers	Overloaded.
setChanged	Sets the status of the current Observable object as "changed." This will cause the hasChanged method to retu rn true.
toString	Overridden. Returns the string representation of the current Observable object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Observable Class

Concepts

java.util Package

Observable Constructor

Constructs a new Observable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Observable();
```

Example

See the example on Observable.

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable Methods

Public Methods

Name	Description
addObserver	Adds an observer to the Observable object, if it is not already in the list.
clearChanged	Causes the method hasChanged to return false, either because the current Observable object has not changed anymore or because the observer has been already notified of the last change.
countObservers	Returns the number of observers for the current Observable object.
deleteObserver	Deletes a specific observer from the list of observers for the current Observable object.
deleteObservers	Deletes all the observers for the current Observable object.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
hasChanged	Checks if the current Observable object has changed.
clone	
notifyObservers	Overloaded.
setChanged	Sets the status of the current Observable object as "changed." This will cause the hasChanged method to return true.
toString	Overridden. Returns the string representation of the current Observable object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Observable Class

Concepts

java.util Package

Observable.addObserver Method

Adds an observer to the Observable object, if it is not already in the list.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized void addObserver(
    java.util.Observer o);
```

Parameters

o

The Observer object to be added.

Example

```
See the example on Observable.
```

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable.clearChanged Method

Causes the method hasChanged to return false, either because the current Observable object has not changed anymore or because the observer has been already notified of the last change.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected synchronized void clearChanged();
```

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable.countObservers Method

Returns the number of observers for the current Observable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized int countObservers();
```

Return Value

The number of observers for the current Observable object.

Example

```
See the example on Observable.
```

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable.deleteObserver Method

Deletes a specific observer from the list of observers for the current Observable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized void deleteObserver(
    java.util.Observer o);
```

Parameters

o

The Observer object to be deleted.

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable.deleteObservers Method

Deletes all the observers for the current Observable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized void deleteObservers();
```

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable.hasChanged Method

Checks if the current Observable object has changed.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean hasChanged();
```

Return Value

true if the object has changed; false otherwise.

Example

```
See the example on Observable.
```

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable.notifyObservers Method

Overload List

Name	Description
Observable.notifyObservers ()	Notifies all the observers of the current Observable object if a change has occurred to this object.
Observable.notifyObservers (Object)	Notifies all the observers of the current Observable object if a change has occurred to this object.

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable.notifyObservers Method ()

Notifies all the observers of the current Observable object if a change has occurred to this object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void notifyObservers();
```

Example

```
See the example on Observable.
```

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable.notifyObservers Method (Object)

Notifies all the observers of the current Observable object if a change has occurred to this object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void notifyObservers(
    java.lang.Object arg);
```

Parameters

arg

An argument describing the change to be notified.

Example

```
See the example on Observable.
```

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observable.setChanged Method

Sets the status of the current Observable object as "changed." This will cause the hasChanged method to return true.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected synchronized void setChanged();
```

Example

```
See the example on Observable.
```

See Also

Reference

Observable Class

Concepts

Observable Members

java.util Package

Observer Interface

An interface that enables the implementing class to be notified of any changes in the observable objects.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.Observer
```

Example

The following example demonstrates the use of most of the methods on the Observable and Observer class. It creates a linked list on the observable object and the changes are sent to the observer object.

```
// Observer-Observable general example

import java.util.*;

class MyClass implements Observer
{
    public void update(Observable o, Object arg)
    {
        Object ob = (LinkedList)arg;
        System.out.println("The list contains: " + ob);
    }

    public static void main(String[] args)
    {
        MyClass mc = new MyClass();
        YourClass yc = new YourClass();
        yc.addObserver(mc);
        System.out.println("Number of observers: " +
                           yc.countObservers());

        // Start watching:
        yc.Watch();
    }
}

class YourClass extends Observable
{
    void Watch()
    {
        // Declare a linked list:
        LinkedList lList = new LinkedList();
        lList.add("Sam");

        // Pass the info the observer:
        setChanged();
        notifyObservers(lList);

        // Check for a change:
        System.out.println("Observable changed?: " + hasChanged());

        // Add nother element:
        lList.add("Pam");

        // Send the new linked list information:
        setChanged();
        notifyObservers(lList);

        // Set the new change:
        setChanged();
```

```
        // Check for a change:
        System.out.println("Observable changed?: " + hasChanged());
    }
}

/*
Output:
Number of observers: 1
The list contains: [Sam]
Observable changed?: false
The list contains: [Sam, Pam]
Observable changed?: true
*/
```

See Also

Concepts

Observer Members

java.util Package

Observer Members

An interface that enables the implementing class to be notified of any changes in the observable objects.

The following tables list the members exposed by the Observer type.

Public Methods

Name	Description
update	Called to update the Observer on a change in the observable object.

See Also

Reference

Observer Interface

Concepts

java.util Package

Observer Methods

Public Methods

Name	Description
update	Called to update the Observer on a change in the observable object.

See Also

Reference

Observer Interface

Concepts

java.util Package

Observer.update Method

Called to update the Observer on a change in the observable object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void update(
    java.util.Observable o,
    java.lang.Object arg);
```

Parameters

o

The observable object.

arg

A parameter passed to the notifyObservers method on the Observable object.

Example

```
See the example on Observer.
```

See Also

Reference

Observer Interface

Concepts

Observer Members

java.util Package

Properties Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class Properties extends Hashtable
```

Inheritance Hierarchy

java.lang.Object

  java.util.Dictionary

    java.util.Hashtable

      java.util.Properties

        java.security.Provider

See Also

Concepts

Properties Members

java.util Package

Properties Members

The following tables list the members exposed by the Properties type.

Public Constructors

Name	Description
Properties	Overloaded.

Public Fields

Name	Description
defaults	

Public Methods

Name	Description
clear	Clears a Hashtable object. (inherited from Hashtable)
contains	Checks for a specific element in a Hashtable object. (inherited from Hashtable)
containsKey	Checks for a specific key in a Hashtable object. (inherited from Hashtable)
containsValue	Checks for a specific value in a Hashtable object. (inherited from Hashtable)
elements	Returns an enumeration that contains the values in a Hashtable object. (inherited from Hashtable)
entrySet	Retrieves the set of mappings contained in a Hashtable object. (inherited from Hashtable)
equals	Overridden.
get	Retrieves an element associated with a specified key in a Hashtable object. (inherited from Hashtable)
hashCode	Returns the hash code of a Hashtable object. (inherited from Hashtable)
getProperty	Overloaded.
getClass	(inherited from Object)
isEmpty	Checks if a Hashtable object is empty. (inherited from Hashtable)
keys	Returns an enumeration of the keys in a Hashtable object. (inherited from Hashtable)
keySet	Returns the key set in a Hashtable object. (inherited from Hashtable)
list	Overloaded.
load	
clone	(inherited from Hashtable)
propertyNames	

put	Adds an element at a specific key entry to a Hashtable object. (inherited from Hashtable)
putAll	Copies all the elements from a specified map to the current Hashtable object. (inherited from Hashtable)
rehash	Rehashes the content of the Hashtable into a larger table. (inherited from Hashtable)
remove	Removes a key/element pair from a Hashtable object. (inherited from Hashtable)
save	Obsolete.
setProperty	
size	Returns the size (number of buckets) of a Hashtable object. (inherited from Hashtable)
store	
toString	Generates the string representation of a Hashtable object. (inherited from Hashtable)
values	Returns the set of elements in a Hashtable object. (inherited from Hashtable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Properties Class

Concepts

java.util Package

Properties Fields

Public Fields

Name	Description
defaults	

See Also

Reference

Properties Class

Concepts

java.util Package

Properties.defaults Field

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.Properties defaults;
```

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties Constructor

Overload List

Name	Description
Properties ()	
Properties (Properties)	

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties Constructor ()

Initializes a new instance of the Properties Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Properties();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties Constructor (Properties)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Properties(
    java.util.Properties defaults);
```

Parameters
defaults

See Also
Reference
Properties Class
Concepts
Properties Members
java.util Package

Properties Methods

Public Methods

Name	Description
clear	Clears a Hashtable object. (inherited from Hashtable)
contains	Checks for a specific element in a Hashtable object. (inherited from Hashtable)
containsKey	Checks for a specific key in a Hashtable object. (inherited from Hashtable)
containsValue	Checks for a specific value in a Hashtable object. (inherited from Hashtable)
elements	Returns an enumeration that contains the values in a Hashtable object. (inherited from Hashtable)
entrySet	Retrieves the set of mappings contained in a Hashtable object. (inherited from Hashtable)
equals	Overridden.
get	Retrieves an element associated with a specified key in a Hashtable object. (inherited from Hashtable)
hashCode	Returns the hash code of a Hashtable object. (inherited from Hashtable)
getProperty	Overloaded.
getClass	(inherited from Object)
isEmpty	Checks if a Hashtable object is empty. (inherited from Hashtable)
keys	Returns an enumeration of the keys in a Hashtable object. (inherited from Hashtable)
keySet	Returns the key set in a Hashtable object. (inherited from Hashtable)
list	Overloaded.
load	
clone	(inherited from Hashtable)
propertyNames	
put	Adds an element at a specific key entry to a Hashtable object. (inherited from Hashtable)
putAll	Copies all the elements from a specified map to the current Hashtable object. (inherited from Hashtable)
rehash	Rehashes the content of the Hashtable into a larger table. (inherited from Hashtable)
remove	Removes a key/element pair from a Hashtable object. (inherited from Hashtable)
save	Obsolete.
setProperty	

size	Returns the size (number of buckets) of a Hashtable object. (inherited from Hashtable)
store	
toString	Generates the string representation of a Hashtable object. (inherited from Hashtable)
values	Returns the set of elements in a Hashtable object. (inherited from Hashtable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Properties Class

Concepts

java.util Package

Properties.equals Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.getProperty Method

Overload List

Name	Description
Properties.getProperty (String)	
Properties.getProperty (String, String)	

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.getProperty Method (String)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getProperty(
    java.lang.String key);
```

Parameters

key

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.getProperty Method (String, String)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getProperty(
    java.lang.String key,
    java.lang.String defaultValue);
```

Parameters

key

defaultValue

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.list Method

Overload List

Name	Description
Properties.list (PrintStream)	
Properties.list (PrintWriter)	

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.list Method (PrintStream)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void list(
    java.io.PrintStream out);
```

Parameters

out

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.list Method (PrintWriter)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void list(
    java.io.PrintWriter out);
```

Parameters

out

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.load Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized void load(
    java.io.InputStream pin) throws java.io.IOException;
```

Parameters

pin

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.propertyNames Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Enumeration propertyNames();
```

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.save Method

NOTE: This Method is now obsolete.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized void save(
    java.io.OutputStream pout,
    java.lang.String header);
```

Parameters

pout

header

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.setProperty Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object setProperty(
    java.lang.String key,
    java.lang.String val);
```

Parameters

key

val

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

Properties.store Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized void store(
    java.io.OutputStream pout,
    java.lang.String header) throws java.io.IOException;
```

Parameters

pout

header

See Also

Reference

Properties Class

Concepts

Properties Members

java.util Package

PropertyResourceBundle Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.PropertyResourceBundle
    extends java.util.ResourceBundle
```

Inheritance Hierarchy

java.lang.Object

  java.util.ResourceBundle

   java.util.PropertyResourceBundle

See Also

Concepts

PropertyResourceBundle Members

java.util Package

PropertyResourceBundle Members

The following tables list the members exposed by the PropertyResourceBundle type.

Public Constructors

Name	Description
PropertyResourceBundle	

Public Fields

Name	Description
parent	(inherited from **ResourceBundle**)

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getKeys	Overridden.
getObject	(inherited from **ResourceBundle**)
getString	(inherited from **ResourceBundle**)
getStringArray	(inherited from **ResourceBundle**)
getClass	(inherited from Object)
handleGetObject	Overridden.
clone	(inherited from **ResourceBundle**)
setParent	(inherited from **ResourceBundle**)
toString	(inherited from **ResourceBundle**)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PropertyResourceBundle Class

Concepts

java.util Package

PropertyResourceBundle Fields

Public Fields

Name	Description
parent	(inherited from **ResourceBundle**)

See Also

Reference

PropertyResourceBundle Class

Concepts

java.util Package

PropertyResourceBundle Constructor

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.PropertyResourceBundle(
    java.io.InputStream stream) throws java.io.IOException;
```

Parameters

stream

See Also

Reference

PropertyResourceBundle Class

Concepts

PropertyResourceBundle Members

java.util Package

PropertyResourceBundle Methods

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
getKeys	Overridden.
getObject	(inherited from **ResourceBundle**)
getString	(inherited from **ResourceBundle**)
getStringArray	(inherited from **ResourceBundle**)
getClass	(inherited from Object)
handleGetObject	Overridden.
clone	(inherited from **ResourceBundle**)
setParent	(inherited from **ResourceBundle**)
toString	(inherited from **ResourceBundle**)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

PropertyResourceBundle Class

Concepts

java.util Package

PropertyResourceBundle.getKeys Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Enumeration getKeys();
```

See Also

Reference

PropertyResourceBundle Class

Concepts

PropertyResourceBundle Members

java.util Package

PropertyResourceBundle.handleGetObject Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final java.lang.Object handleGetObject(
    java.lang.String key);
```

Parameters

key

See Also

Reference

PropertyResourceBundle Class

Concepts

PropertyResourceBundle Members

java.util Package

Random Class

Contains methods to obtain various sorts of random data, such as integers, floats, and other primitives.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.Random
    extends java.lang.Object
    implements java.io.Serializable, System.Runtime.Serialization.ISerializable
```

Example

The following example shows how to generate random values of type boolean, byte[], double, float, Gaussian, int, and long.

```
// random_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        Random r = new Random();

        // Initialize the random number generator using the
        // time from above.
        r.setSeed(time);

        // Generate various types of random data.
        System.out.println("boolean: " + r.nextBoolean());
        byte[] bytes = new byte[5];
        r.nextBytes(bytes);
        System.out.print("byte[]: ");
        for (int j = 0; j < bytes.length; j++)
        {
            System.out.print(bytes[j] + " ");
        }
        System.out.println();
        System.out.println("double: " + r.nextDouble());
        System.out.println("float: " + r.nextFloat());
        System.out.println("Gaussian: " + r.nextGaussian());
        System.out.println("int: " + r.nextInt());
        System.out.println("long: " + r.nextLong());
    }
}

/*
Output:
boolean: false
byte[]: 100 127 20 91 39
double: 0.506746074738395
float: 0.174197257
Gaussian: -1.1872467339778563
int: -722563198
long: -4822180756670951982
*/
```

Inheritance Hierarchy

java.lang.Object

  java.util.Random

    java.security.SecureRandom

See Also

Concepts

Random Members

java.util Package

Random Members

Contains methods to obtain various sorts of random data, such as integers, floats, and other primitives.

The following tables list the members exposed by the Random type.

Public Constructors

Name	Description
Random	Overloaded. Initializes a new instance of a Random object.

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
GetObjectData	Serializes the members of a Random object.
getClass	(inherited from Object)
clone	Creates a new instance of a Random object that is a shallow copy of an existing Random object.
next	Generates a random integer containing the number of bits specified.
nextBoolean	Generates a random boolean value.
nextBytes	Generates a random array of signed bytes.
nextDouble	Generates a random double value between 0 and 1.
nextFloat	Generates a random float value between 0 and 1.
nextGaussian	Generates a random Gaussian value.
nextInt	Overloaded. Generates a random int value.
nextLong	Generates a random long value.
setSeed	Sets the seed used to initialize the random number generator.
toString	Overridden. Displays a human readable summary of a Random object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Random Class

Concepts

java.util Package

Random Constructor

Initializes a new instance of a Random object.

Overload List

Name	Description
Random ()	Initializes a new instance of a Random object.
Random (long)	Initializes a new instance of a Random object with the given seed.
Random (SerializationInfo, StreamingContext)	Initializes a new instance of a Random object during deserialization.

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random Constructor ()

Initializes a new instance of a Random object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Random();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random Constructor (Int64)

Initializes a new instance of a Random object with the given seed.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Random(
    long seed);
```

Parameters

seed

The seed used to initialize the random number generator.

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random Constructor (SerializationInfo, StreamingContext)

Initializes a new instance of a Random object during deserialization.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.Random(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

The information needed to serialize an object.

context

The source or destination for the serialization stream.

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random Methods

Public Methods

Name	Description
equals	(inherited from Object)
hashCode	(inherited from Object)
GetObjectData	Serializes the members of a Random object.
getClass	(inherited from Object)
clone	Creates a new instance of a Random object that is a shallow copy of an existing Random object.
next	Generates a random integer containing the number of bits specified.
nextBoolean	Generates a random boolean value.
nextBytes	Generates a random array of signed bytes.
nextDouble	Generates a random double value between 0 and 1.
nextFloat	Generates a random float value between 0 and 1.
nextGaussian	Generates a random Gaussian value.
nextInt	Overloaded. Generates a random int value.
nextLong	Generates a random long value.
setSeed	Sets the seed used to initialize the random number generator.
toString	Overridden. Displays a human readable summary of a Random object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Random Class

Concepts

java.util Package

Random.GetObjectData Method

Serializes the members of a Random object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void GetObjectData(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

Contains all the information used to serialize an object.

context

Specifies the source or destination for the serialization process.

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.next Method

Generates a random integer containing the number of bits specified.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected synchronized int next(
    int bits);
```

Parameters

bits

The length of the integer to be generated in bits. This value must be greater than or equal to 1 and less than or equal to 32.

Return Value

A random integer between 0 and 2^bits -1.

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.nextBoolean Method

Generates a random boolean value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean nextBoolean();
```

Return Value

A random boolean value (either true or false).

Example

The following example prints out 10 random boolean values.

```
// random_nextboolean.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        // Initialize the random number generator using the
        // time from above.
        Random r = new Random(time);

        // Generate 10 random Boolean values.
        for (int i = 0; i < 10; i++)
        {
            System.out.println(r.nextBoolean());
        }
    }
}

/*
Output:
false
true
true
true
false
false
true
false
true
true
*/
```

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.nextBytes Method

Generates a random array of signed bytes.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void nextBytes(
    byte[] bytes);
```

Parameters

bytes

The array to be populated with random signed bytes.

Example

The following example prints out 10 random arrays of byte values.

```
// random_nextbytes.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        // Initialize the random number generator using the
        // time from above.
        Random r = new Random(time);

        // Generate 10 random arrays of 5 bytes.
        byte[] bytes = new byte[5];
        for (int i = 0; i < 10; i++)
        {
            r.nextBytes(bytes);
            for (int j = 0; j < bytes.length; j++)
            {
                System.out.print(bytes[j] + " ");
            }
            System.out.println();
        }
    }
}

/*
Output:
70 -21 72 -85 78
111 57 -82 -78 -87
99 -104 -5 -62 81
59 61 75 38 -30
-14 45 -46 -36 -39
83 -11 -80 123 33
4 -50 -122 -121 -128
-65 66 -32 76 -93
48 -5 40 69 -15
-80 6 -26 -25 51
*/
```

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.nextDouble Method

Generates a random double value between 0 and 1.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public double nextDouble();
```

Return Value

A random double value between 0 and 1.

Example

The following example prints out 10 random double values.

```
// random_nextdouble.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        // Initialize the random number generator using the
        // time from above.
        Random r = new Random(time);

        // Generate 10 random doubles.
        for (int i = 0; i < 10; i++)
        {
            System.out.println(r.nextDouble());
        }
    }
}

/*
Output:
0.4198760781716504
0.47874095431748742
0.95940806165933323
0.20391668127336116
0.53872371526715213
0.57789881865073889
0.36308423397158118
0.30986392774496274
0.815240298456572
0.29822636217031273
*/
```

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.nextFloat Method

Generates a random float value between 0 and 1.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public float nextFloat();
```

Return Value

A random float value between 0 and 1.

Example

The following example prints out 10 random float values.

```
// random_nextfloat.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        // Initialize the random number generator using the
        // time from above.
        Random r = new Random(time);

        // Generate 10 random floats.
        for (int i = 0; i < 10; i++)
        {
            System.out.println(r.nextFloat());
        }
    }
}

/*
Output:
0.6807229
0.271337152
0.0456498861
0.424616158
0.145857215
0.313717842
0.793788552
0.0299280882
0.9014773
0.0284169316
*/
```

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.nextGaussian Method

Generates a random Gaussian value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized double nextGaussian();
```

Return Value

A random Gaussian value.

Example

The following example prints out 10 random Gaussian values.

```
// random_nextgaussian.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        // Initialize the random number generator using the
        // time from above.
        Random r = new Random(time);

        // Generate 10 random Gaussian values.
        for (int i = 0; i < 10; i++)
        {
            System.out.println(r.nextGaussian());
        }
    }
}

/*
Output:
0.14357387227838234
-0.046254647058384474
1.8878794717808312
-0.453012430053396
-1.3389395194956137
-1.5308781476352151
-1.4203675170816268
-1.9396826969279308
0.55111333173309363
0.9477179313359444
*/
```

Remarks

The nextGaussian method generates Gaussian numbers with a mean of 0.0 and a standard deviation of 1.0. The algorithm is based on the polar form of the Box-Muller Transformation applied to uniformly distributed random numbers.

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.nextInt Method

Generates a random int value.

Overload List

Name	Description
Random.nextInt ()	Generates a random int value.
Random.nextInt (int)	Generates a random int value between 0 and n.

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.nextInt Method ()

Generates a random int value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int nextInt();
```

Return Value

A random int value.

Example

The following example prints out 10 random int values.

```
// random_nextint.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        // Initialize the random number generator using the
        // time from above.
        Random r = new Random(time);

        // Generate 10 random ints.
        for (int i = 0; i < 10; i++)
        {
            System.out.println(r.nextInt());
        }
    }
}

/*
Output:
-694884771
-599535625
1085797709
-713459955
-634827481
-1861079132
-972042854
-1974631783
-1800648982
-195619644
*/
```

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.nextInt Method (Int32)

Generates a random int value between 0 and n.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int nextInt(
    int n);
```

Parameters

n

The maximum value for the int to be generated. This value must be between 0 and java.lang.Integer.MAX_INTEGER.

Return Value

A random int value between 0 and n.

Example

The following example prints out 10 random int values less than 1,000,000.

```
// random_nextint_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        // Initialize the random number generator using the
        // time from above.
        Random r = new Random(time);

        // Generate 10 random ints less than 1,000,000.
        for (int i = 0; i < 10; i++)
        {
            System.out.println(r.nextInt(1000000));
        }
    }
}

/*
Output:
285684
798051
635022
148892
525169
461856
908004
282498
834215
987625
*/
```

See Also

Reference

Random Class
Concepts
Random Members
java.util Package

Random.nextLong Method

Generates a random long value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public long nextLong();
```

Return Value

A random long value.

Example

The following example prints out 10 random long values.

```
// random_nextlong.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        // Initialize the random number generator using the
        // time from above.
        Random r = new Random(time);

        // Generate 10 random longs.
        for (int i = 0; i < 10; i++)
        {
            System.out.println(r.nextLong());
        }
    }
}

/*
Output:
2768182117229783821
3837250559113071772
-8017095318783076180
-8447983171381820751
-1501838064363879837
787972653359279082
-4721216399848929119
1444225638129320913
6145099151853620487
4389531239905850113
*/
```

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Random.setSeed Method

Sets the seed used to initialize the random number generator.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized void setSeed(
    long seed);
```

Parameters

seed

The seed used to initialize the random number generator.

Example

The following example shows how to initialize the random number generator using the current time.

```
// random_setseed.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Get the current time.
        Date date = new Date();
        long time = date.getTime();

        Random r = new Random();

        // Initialize the random number generator using the
        // time from above.
        r.setSeed(time);
    }
}
```

See Also

Reference

Random Class

Concepts

Random Members

java.util Package

Set Interface

Provides a set of objects that implements the Collection interface.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.Set
    extends java.util.Collection
```

Remarks

This interface is implemented by collection classes such as HashSet, TreeSet.

See Also

Concepts

Set Members

java.util Package

Set Members (J#)

Provides a set of objects that implements the Collection interface.

The following tables list the members exposed by the Set type.

Public Methods

Name	Description
add	Adds the given element to a Set and returns false if the object can not be added.
addAll	Adds all elements from given collection into a Set.
clear	Removes all the elements from a Set object.
contains	Checks if a Set object contains a specific element.
containsAll	Checks if a Set object contains all the elements in a specified collection.
equals	Checks for equality between a specified element and the current element in a Set object.
hashCode	Retrieves the hash code for a Set object.
isEmpty	Checks if a Set object is empty.
iterator	Provides an iterator over a Set object.
remove	Deletes an element from a Set object.
removeAll	Deletes all elements of a specified collection from a Set object.
retainAll	Deletes all elements of the Set that are not contained in specified collection.
size	Retrieves the number of elements in a Set.
toArray	Overloaded. Converts the Set object to an array.

See Also

Reference

Set Interface

Concepts

java.util Package

Set Methods

Public Methods

Name	Description
add	Adds the given element to a Set and returns false if the object can not be added.
addAll	Adds all elements from given collection into a Set.
clear	Removes all the elements from a Set object.
contains	Checks if a Set object contains a specific element.
containsAll	Checks if a Set object contains all the elements in a specified collection.
equals	Checks for equality between a specified element and the current element in a Set object.
hashCode	Retrieves the hash code for a Set object.
isEmpty	Checks if a Set object is empty.
iterator	Provides an iterator over a Set object.
remove	Deletes an element from a Set object.
removeAll	Deletes all elements of a specified collection from a Set object.
retainAll	Deletes all elements of the Set that are not contained in specified collection.
size	Retrieves the number of elements in a Set.
toArray	Overloaded. Converts the Set object to an array.

See Also

Reference

Set Interface

Concepts

java.util Package

Set.add Method

Adds the given element to a Set and returns false if the object can not be added.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean add(
    java.lang.Object e);
```

Parameters

e

The object to be added to the Set.

Return Value

true if the object was successfully added; false otherwise.

Example

In this example, you create a HashSet object, which implements the Set interface. Then you add some elements to the set and display it.

```
// Set-add1.jsl
// Set.add example
import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashSet object:
        Set hs = new HashSet();

        Integer x = new Integer(22);
        Double y = new Double(2.45);

        // Add some elements to the HashSet object:
        System.out.println(hs.add(x) + ", " + hs.add(y));

        // Display the HashSet object:
        System.out.println(hs);
   }
}

/*
Output
true, true
[2.45,22]
*/
```

Remarks

Throws UnSupportedOperationException if this method is not supported by the implementing class of this interface.

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.addAll Method

Adds all elements from given collection into a Set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean addAll(
    java.util.Collection c);
```

Parameters

c

The name of the collection.

Return Value

true if the elements were inserted successfully; false otherwise.

Example

In this example, you declare a HashSet object, hs, object and a LinkedList object, ll. You then add some elements to ll and copy it to hs. When you display hs, you get the same elements as those of ll.

```
// Set-addA1.jsl
// Set.addAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashSet and LinkedList objects:
        Set hs = new HashSet();
        LinkedList ll =  new LinkedList();

        // Add some elements to hs:
        ll.add("My Node");
        ll.add("Your Node");
        ll.add(null);
        ll.add(new Integer(55));

        // Copy ll to hs and display hs:
        if(hs.addAll(ll)==true)
            System.out.println(hs);
    }
}

/*
Output:
[null, My Node, 55, Your Node]
*/
```

Remarks

Throws UnsupportedOperationException if this method is not supported by the implementing class of this interface.

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.clear Method

Removes all the elements from a Set object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void clear();
```

Example

In this example, you create a hash set, add some elements to it and then clear it. You display the set before and after clearing.

```
// Set-clear.jsl
// Set.clear example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashSet object:
        Set hs = new HashSet();

        Integer x = new Integer(22);
        Double y = new Double(2.45);

        // Add some elements:
        hs.add(x);
        hs.add(y);

        // Display the HashSet object:
        System.out.println(hs);

        // Clear the HashSet:
        hs.clear();

        // Display the HashSet object after clearing:
        System.out.println(hs);

   }
}

/*
Output
[22, 2.45]
[]
*/
```

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.contains Method

Checks if a Set object contains a specific element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean contains(
    java.lang.Object e);
```

Parameters

e

The element checked for.

Return Value

true if the object exists in Set; false otherwise.

Example

In this example, you create and initialize a hash set object, and then you check to see if the set contains a specified element.

```
// Set-con1.jsl
// Set.contains example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashSet object:
        Set hs = new HashSet();

        // Add some elements:
        hs.add("My Node");
        hs.add("Your Node");
        hs.add(null);
        hs.add(new Integer(55));

        // Check if "My Node" is in the list:
        if(hs.contains("My Node"))
            System.out.println(
                      "The object \"My Node\" is in the list.");
    }
}

/*
Output:
The object "My Node" is in the list.
*/
```

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.containsAll Method

Checks if a Set object contains all the elements in a specified collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean containsAll(
    java.util.Collection c);
```

Parameters

c

The name of the collection.

Return Value

true if the elements were found in the Set object; false otherwise.

Example

In this example, you create a linked list object and add some elements to it. Then you create a hash set object and copy the linked list object to it. Testing the hash set object to see if it contains all the elements of the linked list gives the result true.

```
// Set-contAll.jsl
// Set.containsAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList lList =  new LinkedList();

        // Add some elements:
        lList.add("Sam");
        lList.add("Michelle");
        lList.add("Jim");
        lList.add("Virginia");

        // Create a HashSet object:
        Set hs = new HashSet();

        // Copy lList elements to hs:
        hs.addAll(lList);

        // Check if has contains all the lelemnts of lList:
        if(hs.containsAll(lList))
            System.out.println("The two objects are identical.");
        System.out.println("hs = " + hs);
    }
}

/*
Output:
The two objects are identical.
hs = [Michelle, Jim, Sam, Virginia]
*/
```

See Also

Reference

Set Interface
Concepts
Set Members
java.util Package

Set.equals Method

Checks for equality between a specified element and the current element in a Set object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean equals(
    java.lang.Object e);
```

Parameters

e

The element compared with.

Return Value

true if the two elements are identical; false otherwise.

Example

In this example, you create two hash set objects, add some elements to the first one, and copy it to the second one. Then you test the two objects for equality.

```
// Set-eq1.jsl
// Set.Equals example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create two HashSet objects:
        Set hs1 = new HashSet();
        Set hs2 = new HashSet();

        // Add some elements:
        hs1.add("Sam");
        hs1.add("Michelle");
        hs1.add("Jim");
        hs1.add("Virginia");

        // Add all the element of hs1 to hs11:
        hs2.addAll(hs1);

        // Check if the two objects are identical:
        if(hs1.Equals(hs2))
            System.out.println("The two objects are the same.");
    }
}

/*
Output:
The two objects are the same.
*/
```

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.hashCode Method

Retrieves the hash code for a Set object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int hashCode();
```

Return Value

The integer that represents the hash code of the Set object.

Example

In this example, you display the hash code of a hash set object after adding some elements to it.

```
// Set-getHash1.jsl
// Set.GetHashCode example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashSet object:
        Set hs = new HashSet();

        // Add some elements:
        hs.add(new Double(5.25));
        hs.add(new Byte((byte)127));
        hs.add(null);
        hs.add(new Integer(55));

        // Print the hash code:
        System.out.println("The hash code is: " + hs.GetHashCode());
    }
}

/*
The hash code is: 1075118262
*/
```

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.isEmpty Method

Checks if a Set object is empty.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean isEmpty();
```

Return Value

true if the Set object was empty; false otherwise.

Example

In this example, you create and initialize a hash set object, clear it, and then you check to see if the set is empty.

```
// Set-isEmp1.jsl
// Set.isEmpty example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashSet object:
        Set hs = new HashSet();

        // Add some elements:
        hs.add("Hazem");
        hs.add("Sally");
        hs.add("Pille");
        hs.add("Craig");

        // Clear the list:
        hs.clear();

        // Check if the list is empty:
        if(hs.isEmpty())
            System.out.println("The list is now empty.");

        // Display the empty set:
        System.out.println("hs = " + hs);
    }
}

/*
The list is now empty.
hs = []
*/
```

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.iterator Method

Provides an iterator over a Set object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Iterator iterator();
```

Return Value

The provided iterator over the Set object.

Example

In this example, you create an initialize a hash set object and then you retrieve an iterator to iterate over the set and display its members.

```
// Set-iterator1.jsl
// Set.iterator example

import java.util.*;

public class MyClass
{
   public static void main()
   {
      // Create a HashSet object:
      Set hs = new HashSet();

      // Add some elements:
      hs.add("1 ");
      hs.add("2 ");
      hs.add("3 ");
      hs.add("4 ");

      // Retrieve an iterator to the hashset:
      Iterator iter = hs.iterator();

      // Extract elements from iterator.
      // Note that the elements may not follow the order in which they
      // are added to HashSet.
      while(iter.hasNext())
      {
         System.out.print(iter.next());
         iter.remove();
      }
   }
}
/*
Output:
3 1 4 2
*/
```

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.remove Method

Deletes an element from a Set object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean remove(
    java.lang.Object e);
```

Parameters

e

The element to be removed from the Set object.

Return Value

true if the element was successfully removed; false otherwise.

Example

In this example, you create and initialize a hash set object, then you remove one element and display the set before and after the removal.

```
// Set-rem1.jsl
// Set.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashSet object:
        Set hs = new HashSet();

        // Add some elements:
        hs.add("Sam");
        hs.add("Michelle");
        hs.add("Jim");
        hs.add("Virginia");

        // Display the set:
        System.out.println("hs (before) = " + hs);

        // Remove an element:
        hs.remove("Sam");

        // Display the set again:
        System.out.println("hs (after) = " + hs);
   }
}

/*
Output:
hs (before) = [Michelle, Jim, Sam, Virginia]
hs (after) = [Michelle, Jim, Virginia]
*/
```

See Also

Reference

Set Interface

Concepts

Set Members
java.util Package

Set.removeAll Method

Deletes all elements of a specified collection from a Set object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean removeAll(
    java.util.Collection c);
```

Parameters

c

The name of the collection.

Return Value

true if the elements were removed successfully; false otherwise.

Example

In this example, you create and initialize two collections, a linked list (ll) and a hash set (hs). Then you remove all the elements from hs that exist in ll. The output displays the new contents of hs.

```
// Set-remA1.jsl
// Set.removeAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList ll =  new LinkedList();

        // Create a HashSet object:
        Set hs = new HashSet();

        // Add some elements to hs:
        hs.add("Isabella");
        hs.add("Angelina");
        hs.add("Pille");
        hs.add("Hazem");

        // Add some elements to ll:
        ll.add("Isabella");
        ll.add("Angelina");

        // Remove ll members:
        hs.removeAll(ll);

        // Display hs:
        System.out.println("The collection is: " + hs);
    }
}

/*
Output:
The collection is: [Pille, Hazem]
*/
```

See Also

Reference
Set Interface
Concepts
Set Members
java.util Package

Set.retainAll Method

Deletes all elements of the Set that are not contained in specified collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean retainAll(
    java.util.Collection c);
```

Parameters

c

The name of the collection.

Return Value

true if the elements were deleted successfully; false otherwise.

Example

In this example, you create and initialize two collections, a linked list (ll) and a hash set (hs). Then you retain all the elements of ll into hs. The output displays the contents of both collections.

```
// Set-retA1.jsl
// Set.retainAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList ll =  new LinkedList();

        // Create a HashSet object:
        Set hs = new HashSet();

        // Add some elements to hs:
        hs.add("Isabella");
        hs.add("Angelina");
        hs.add("Pille");
        hs.add("Hazem");

        // Add some elements to ll:
        ll.add("Isabella");
        ll.add("Angelina");

        // Retain elements from ll into hs:
        hs.retainAll(ll);

        // Display hs:
        System.out.println("hs contains: " + hs);
        System.out.println("ll contains: " + ll);
    }
}

/*
Output:
hs contains: [Isabella, Angelina]
ll contains: [Isabella, Angelina]
*/
```

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.size Method

Retrieves the number of elements in a Set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int size();
```

Return Value

An integer that represents the number of elements in the Set object.

Example

In this example, you create and initialize a hash set object, then you clear the elements and display the size before and after the clearing.

```
// Set-size.jsl
// Set.size example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a HashSet object:
        Set hs = new HashSet();

        // Add some elements:
        hs.add("100");
        hs.add("200");
        hs.add("300");
        hs.add("400");

        // Get and display the elements:
        System.out.print("The elements of the set are: ");
        System.out.println(hs);

        // Print the size:
        System.out.println("The size of the set is: " + hs.size());

        // Clear the set:
        hs.clear();
        System.out.print("The set has been cleared. ");

        // Print the new size:
        System.out.println("The size is now: " + hs.size());
    }
}

/*
Output:
The elements of the set are: [100, 300, 200, 400]
The size of the set is: 4
The set has been cleared. The size is now: 0
*/
```

See Also

Reference

Set Interface

Concepts

Set Members
java.util Package

Set.toArray Method

Converts the Set object to an array.

Overload List

Name	Description
Set.toArray ()	Converts the Set object to an array.
Set.toArray (Object[])	Fills the specified array with the elements of Set object.

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.toArray Method ()

Converts the Set object to an array.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object[] toArray();
```

Return Value

The array object that contains the elements of the Set object.

Example

In this example, you create and initialize a hash set object, and then you convert the set to an array and display the elements of the array.

```
// Set-toArr1.jsl
// Set.toArray example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList hs =  new LinkedList();

        // Add some elements:
        hs.add("Sarah");
        hs.add("Isabella");
        hs.add("Angelina");
        hs.add("Bianca");

        // Create an array from the list:
        Object[] s = hs.toArray();

        // Display the array elements:
        System.out.println("The array elements are:");
        for (int i=0; i<hs.size(); i++)
            System.out.println("Element #" + i + " = " + s[i]);
    }
}

/*
Output:
The array elements are:
Element #0 = Sarah
Element #1 = Isabella
Element #2 = Angelina
Element #3 = Bianca
*/
```

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

Set.toArray Method (Object[])

Fills the specified array with the elements of Set object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object[] toArray(
    java.lang.Object[] arr);
```

Parameters

arr

The name of the target array.

Return Value

The array object that contains the elements of the Set object.

Example

In this example, you create and initialize a hash set object, and then you convert the set to an array "s". Then you copy "s" to a new array "s1" and display the elements of "s1."

```
// Set-toArr2.jsl
// Set.toArray example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a linked list object:
        LinkedList hs =  new LinkedList();

        // Add some elements:
        hs.add("Sarah");
        hs.add("Isabella");
        hs.add("Angelina");
        hs.add("Bianca");

        // Create an array from the list:
        Object[] s = hs.toArray();

        // Copy "s" to another object array "s1":
        Object[] s1 = hs.toArray(s);


        // Display the elements of the new array:
        System.out.println("The array elements are:");
        for (int i=0; i<hs.size(); i++)
            System.out.println("Element #" + i + " = " + s1[i]);
    }
}

/*
Output:
The array elements are:
Element #0 = Sarah
Element #1 = Isabella
Element #2 = Angelina
Element #3 = Bianca
*/
```

Remarks

The element at the end of the data (at the index Set.size()) is set to null if the given array has more elements than size of Set.

Copies only the elements that array can store if the array is smaller than size of Set.

See Also

Reference

Set Interface

Concepts

Set Members

java.util Package

SimpleTimeZone Class

Represents a specialization of the TimeZone class containing methods useful for calculations involving time zones, such as offsets from UTC and determining whether daylight savings time is in effect.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.SimpleTimeZone
    extends java.util.TimeZone
```

Example

The following example demonstrates the getRawOffset, inDaylightTime, setEndRule, setStartRule, and setStartYear methods of the SimpleTimeZone class.

```
// simpletimezone_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create our own time zone named XYZ with an offset of
        // 100 milliseconds from GMT.  Set the daylight savings
        // time period from the first Sunday in April to the last
        // Sunday in October.
        SimpleTimeZone xyz = new SimpleTimeZone(100, "XYZ",
            Calendar.APRIL, 1, Calendar.SUNDAY, 2,
            Calendar.OCTOBER, -1, Calendar.SUNDAY, 2);
        TimeZone.setDefault(xyz);

        // Change the daylight savings time rules effective in the
        // year 2005.
        xyz.setStartYear(105);

        // Change the starting time for daylight savings time to
        // the first Sunday in July at 2:00 AM.
        xyz.setStartRule(Calendar.JULY, 1, Calendar.SUNDAY, 2);

        // Change the ending time for daylight savings time to
        // the last Sunday in July at 2:00 AM.
        xyz.setEndRule(Calendar.JULY, -1, Calendar.SUNDAY, 2);

        // Determine if a few dates are in daylight savings time.
        Date d1 = new Date(105, Calendar.APRIL, 15, 00, 00, 00);
        Date d2 = new Date(105, Calendar.JULY, 15, 00, 00, 00);
        Date d3 = new Date(105, Calendar.DECEMBER, 31, 00, 00, 00);

        boolean isD1InDaylight = xyz.inDaylightTime(d1);
        boolean isD2InDaylight = xyz.inDaylightTime(d2);
        boolean isD3InDaylight = xyz.inDaylightTime(d3);

        System.out.println("d1 is in DST? " + isD1InDaylight);
        System.out.println("d2 is in DST? " + isD2InDaylight);
        System.out.println("d3 is in DST? " + isD3InDaylight);

        // Display the raw offset for this time zone.
        int offset = xyz.getRawOffset();
        System.out.println("raw offset: " + offset);
    }
}
```

```
/*
Output:
d1 is in DST? false
d2 is in DST? true
d3 is in DST? false
raw offset: 100
*/
```

Inheritance Hierarchy

java.lang.Object

  java.util.TimeZone

   java.util.SimpleTimeZone

See Also

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone Members

Represents a specialization of the TimeZone class containing methods useful for calculations involving time zones, such as offsets from UTC and determining whether daylight savings time is in effect.

The following tables list the members exposed by the SimpleTimeZone type.

Public Constructors

Name	Description
SimpleTimeZone	Overloaded. Initializes a new instance of a SimpleTimeZone object.

Public Methods

Name	Description
equals	Overridden. Determines whether two SimpleTimeZone objects are equal.
hashCode	Overridden. Generates a hash value representing the contents of the object.
getID	Gets the ID of the current time zone. (inherited from TimeZone)
getOffset	Overridden. Gets the offset of the time zone from UTC of the given date, including any effects of daylight savings time.
getRawOffset	Overridden. Gets the offset of the time zone from UTC, ignoring daylight savings time.
getClass	(inherited from Object)
inDaylightTime	Overridden. Determines whether daylight savings time is in effect on the given date.
clone	Creates a new instance of a SimpleTimeZone object that is a shallow copy of the current object.
setEndRule	Sets the rule for when daylight savings time comes to an end.
setID	Sets the ID used to represent the current time zone. (inherited from TimeZone)
setRawOffset	Overridden. Sets the offset of the time zone from UTC, ignoring daylight savings time.
setStartRule	Sets the rule for when daylight savings time begins.
setStartYear	Sets the year when the rules governing this time zone begin.
toString	Displays a human-readable representation of a TimeZone object. (inherited from TimeZone)
useDaylightTime	Overridden. Determines whether the current time zone uses daylight savings time.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Constructor

Initializes a new instance of a SimpleTimeZone object.

Overload List

Name	Description
SimpleTimeZone (int, String)	Initializes a new instance of a SimpleTimeZone object with the given raw offset from UTC and ID.
SimpleTimeZone (int, String, int, int, int, int, int, int, int, int)	Initializes a new instance of a SimpleTimeZone object with the given raw offset from UTC, ID, and start and end values for daylight saving s time.

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone Constructor (Int32, String)

Initializes a new instance of a SimpleTimeZone object with the given raw offset from UTC and ID.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.SimpleTimeZone(
    int rawOffset,
    java.lang.String ID);
```

Parameters

rawOffset

The offset, represented in milliseconds, of the time zone from UTC, ignoring daylight savings time. Areas east of UTC have positive offsets, while areas west of UTC have negative offsets.

ID

The ID used to represent this time zone.

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone Constructor (Int32, String, Int32, Int32, Int32, Int32, Int32, Int32, Int32, Int32)

Initializes a new instance of a SimpleTimeZone object with the given raw offset from UTC, ID, and start and end values for daylight savings time.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.SimpleTimeZone(
    int rawOffset,
    java.lang.String ID,
    int startMonth,
    int startDayOfWeekInMonth,
    int startDayOfWeek,
    int startTime,
    int endMonth,
    int endDayOfWeekInMonth,
    int endDayOfWeek,
    int endTime);
```

Parameters

rawOffset

The offset, represented in milliseconds, of the time zone from UTC, ignoring daylight savings time. Areas east of UTC have positive offsets, while areas west of UTC have negative offsets.

ID

The ID used to represent this time zone.

startMonth

The month when daylight savings time begins, such as APRIL.

startDayOfWeekInMonth

The day of the week in the month when daylight savings time begins, such as 4 to represent the fourth Sunday of the month. Use -1 to mean the last day of the week in the month.

startDayOfWeek

The day when daylight savings time begins, such as SUNDAY.

startTime

The time when daylight savings time begins, in hours. For example, use 2 to represent 2:00 AM.

endMonth

The month when daylight savings time ends, such as OCTOBER.

endDayOfWeekInMonth

The day of the week in the month when daylight savings time ends, such as 4 to represent the fourth Sunday of the month. Use -1 to mean the last day of the week in the month.

endDayOfWeek

The day when daylight savings time ends, such as SUNDAY.

endTime

The time when daylight savings time ends, in hours. For example, use 2 to represent 2:00 AM.

See Also

Reference

SimpleTimeZone Class
Concepts
SimpleTimeZone Members
java.util Package

SimpleTimeZone Methods

Public Methods

Name	Description
equals	Overridden. Determines whether two SimpleTimeZone objects are equal.
hashCode	Overridden. Generates a hash value representing the contents of the object.
getID	Gets the ID of the current time zone. (inherited from TimeZone)
getOffset	Overridden. Gets the offset of the time zone from UTC of the given date, including any effects of daylight savings time.
getRawOffset	Overridden. Gets the offset of the time zone from UTC, ignoring daylight savings time.
getClass	(inherited from Object)
inDaylightTime	Overridden. Determines whether daylight savings time is in effect on the given date.
clone	Creates a new instance of a SimpleTimeZone object that is a shallow copy of the current object.
setEndRule	Sets the rule for when daylight savings time comes to an end.
setID	Sets the ID used to represent the current time zone. (inherited from TimeZone)
setRawOffset	Overridden. Sets the offset of the time zone from UTC, ignoring daylight savings time.
setStartRule	Sets the rule for when daylight savings time begins.
setStartYear	Sets the year when the rules governing this time zone begin.
toString	Displays a human-readable representation of a TimeZone object. (inherited from TimeZone)
useDaylightTime	Overridden. Determines whether the current time zone uses daylight savings time.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

SimpleTimeZone Class

Concepts

java.util Package

SimpleTimeZone.clone Method

Creates a new instance of a SimpleTimeZone object that is a shallow copy of the current object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object clone();
```

Return Value

A new instance of a SimpleTimeZone object that is a shallow copy of the current object.

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone.equals Method

Determines whether two SimpleTimeZone objects are equal.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean equals(
    java.lang.Object obj);
```

Parameters

obj

A SimpleTimeZone object to compare with the SimpleTimeZone object calling this method.

Return Value

true if the two SimpleTimeZone objects are equal; false otherwise.

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone.hashCode Method

Generates a hash value representing the contents of the object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized int hashCode();
```

Return Value

A hash value representing the contents of the object.

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone.getOffset Method

Gets the offset of the time zone from UTC of the given date, including any effects of daylight savings time.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getOffset(
    int era,
    int year,
    int month,
    int day,
    int dayOfWeek,
    int milliSeconds);
```

Parameters

era

The era of the date being used to calculate the offset. For the Gregorian calendar system, the value will be one of BC or AD.

year

The year of the date being used to calculate the offset. This value is the current year - 1900.

month

The month of the date being used to calculate the offset. This value is one of the month constants in the Calendar class, such as JANUARY.

day

The day of the month for the date being used to calculate the offset.

dayOfWeek

The day of the week for the date being used to calculate the offset. This value is one of the day constants in the Calendar class, such as SUNDAY.

milliSeconds

The time, in milliseconds, for the date being used to calculate the offset.

Return Value

The offset, represented in milliseconds, of the time zone from UTC, including any effects of daylight savings time. Areas east of UTC will get a positive return value, while areas west of UTC will get a negative return value.

Example

The following example shows the offset for the default time zone for both January 1, 2005 and July 1, 2005. In areas where daylight savings time is observed, the two offsets will be different.

```
// simpletimezone_getoffset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone def = SimpleTimeZone.getDefault();

        // Saturday, January 1, 2005 at midnight.
        int offset1 = def.getOffset(
            GregorianCalendar.AD,
            105,
```

```
            Calendar.JANUARY,
            1,
            Calendar.SATURDAY,
            0);
        System.out.println("January 1 offset: " + offset1);

        // Friday, July 1, 2005 at midnight.
        int offset2 = def.getOffset(
            GregorianCalendar.AD,
            105,
            Calendar.JULY,
            1,
            Calendar.FRIDAY,
            0);
        System.out.println("July 1 offset: " + offset2);
    }
}

/*
Output:
January 1 offset: -28800000
July 1 offset: -25200000
*/
```

See Also
Reference
SimpleTimeZone Class
Concepts
SimpleTimeZone Members
java.util Package

SimpleTimeZone.getRawOffset Method

Gets the offset of the time zone from UTC, ignoring daylight savings time.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int getRawOffset();
```

Return Value

The offset, represented in milliseconds, of the time zone from UTC, ignoring daylight savings time. Areas east of UTC will get a positive return value, while areas west of UTC will get a negative return value.

Example

The following example shows how to get the raw offset for the default time zone. Daylight savings time does not factor into the raw offset.

```
// simpletimezone_getrawoffset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone def = SimpleTimeZone.getDefault();

        int offset = def.getRawOffset();
        System.out.println("Raw offset: " + offset);
    }
}

/*
Output:
Raw offset: -28800000
*/
```

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone.inDaylightTime Method

Determines whether daylight savings time is in effect on the given date.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean inDaylightTime(
    java.util.Date date);
```

Parameters

date

The Date object used to determine if daylight savings time is in effect.

Return Value

true if daylight savings time is in effect on the given date; false otherwise.

Example

The following example shows how to determine if a Date is in daylight savings time.

```
// simpletimezone_indaylighttime.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone def = SimpleTimeZone.getDefault();

        // Saturday, January 1, 2005 at midnight.
        Date d1 = new Date(
            105,
            Calendar.JANUARY,
            1,
            00,
            00,
            00);
        boolean isInDaylight = def.inDaylightTime(d1);
        System.out.println("January 1 in DST? " + isInDaylight);

        // Friday, July 1, 2005 at midnight.
        Date d2 = new Date(
            105,
            Calendar.JULY,
            1,
            00,
            00,
            00);
        isInDaylight = def.inDaylightTime(d2);
        System.out.println("July 1 in DST? " + isInDaylight);
    }
}

/*
Output:
January 1 in DST? false
July 1 in DST? true
*/
```

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone.setEndRule Method

Sets the rule for when daylight savings time comes to an end.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setEndRule(
    int month,
    int dayOfWeekInMonth,
    int dayOfWeek,
    int time);
```

Parameters

month

The month when daylight savings time ends, such as OCTOBER.

dayOfWeekInMonth

The day of the week in the month when daylight savings time ends, such as 4 to represent the fourth Sunday of the month. Use -1 to mean the last day of the week in the month.

dayOfWeek

The day when daylight savings time ends, such as SUNDAY.

time

The time when daylight savings time ends, in hours. For example, use 2 to represent 2:00 AM.

Example

The following example changes the end rule for a made-up time zone to the last Sunday in July at 2:00 AM.

```
// simpletimezone_setendrule.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create our own time zone named XYZ with an offset of
        // 100 milliseconds from GMT.  Set the daylight savings
        // time period from the first Sunday in April to the last
        // Sunday in October.
        SimpleTimeZone xyz = new SimpleTimeZone(100, "XYZ",
            Calendar.APRIL, 1, Calendar.SUNDAY, 2,
            Calendar.OCTOBER, -1, Calendar.SUNDAY, 2);
        TimeZone.setDefault(xyz);

        // Change the starting time for daylight savings time to
        // the first Sunday in July at 2:00 AM.
        xyz.setStartRule(Calendar.JULY, 1, Calendar.SUNDAY, 2);

        // Change the ending time for daylight savings time to
        // the last Sunday in July at 2:00 AM.
        xyz.setEndRule(Calendar.JULY, -1, Calendar.SUNDAY, 2);

        // Determine if a few dates are in daylight savings time.
        Date d1 = new Date(105, Calendar.JULY, 3, 01, 00, 00);
        Date d2 = new Date(105, Calendar.JULY, 3, 03, 00, 00);
        Date d3 = new Date(105, Calendar.DECEMBER, 31, 00, 00, 00);
```

```
        boolean isD1InDaylight = xyz.inDaylightTime(d1);
        boolean isD2InDaylight = xyz.inDaylightTime(d2);
        boolean isD3InDaylight = xyz.inDaylightTime(d3);

        System.out.println("d1 is in DST? " + isD1InDaylight);
        System.out.println("d2 is in DST? " + isD2InDaylight);
        System.out.println("d3 is in DST? " + isD3InDaylight);
    }
}

/*
Output:
d1 is in DST? false
d2 is in DST? true
d3 is in DST? false
*/
```

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone.setRawOffset Method

Sets the offset of the time zone from UTC, ignoring daylight savings time.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setRawOffset(
    int offsetMillis);
```

Parameters

offsetMillis

The offset, represented in milliseconds, of the time zone from UTC, ignoring daylight savings time. Areas east of UTC have positive offsets, while areas west of UTC have negative offsets.

Example

The following example changes the raw offset for a made-up time zone to 200 milliseconds off GMT.

```
// simpletimezone_setrawoffset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create our own time zone named XYZ with an offset of
        // 100 milliseconds from GMT.  Set the daylight savings
        // time period from the first Sunday in April to the last
        // Sunday in October.
        SimpleTimeZone xyz = new SimpleTimeZone(100, "XYZ",
            Calendar.APRIL, 1, Calendar.SUNDAY, 2,
            Calendar.OCTOBER, -1, Calendar.SUNDAY, 2);
        TimeZone.setDefault(xyz);

        // Change the offset to 200 milliseconds.
        xyz.setRawOffset(200);

        System.out.println("raw offset: " + xyz.getRawOffset());
    }
}

/*
Output:
raw offset: 200
*/
```

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone.setStartRule Method

Sets the rule for when daylight savings time begins.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setStartRule(
    int month,
    int dayOfWeekInMonth,
    int dayOfWeek,
    int time);
```

Parameters

month

The month when daylight savings time begins, such as APRIL.

dayOfWeekInMonth

The day of the week in the month when daylight savings time begins, such as 4 to represent the fourth Sunday of the month. Use -1 to mean the last day of the week in the month.

dayOfWeek

The day when daylight savings time begins, such as SUNDAY.

time

The time when daylight savings time begins, in hours. For example, use 2 to represent 2:00 AM.

Example

The following example changes the start rule for a made-up time zone to the first Sunday in July at 2:00 AM.

```
// simpletimezone_setstartrule.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create our own time zone named XYZ with an offset of
        // 100 milliseconds from GMT.  Set the daylight savings
        // time period from the first Sunday in April to the last
        // Sunday in October.
        SimpleTimeZone xyz = new SimpleTimeZone(100, "XYZ",
            Calendar.APRIL, 1, Calendar.SUNDAY, 2,
            Calendar.OCTOBER, -1, Calendar.SUNDAY, 2);
        TimeZone.setDefault(xyz);

        // Change the starting time for daylight savings time to
        // the first Sunday in July at 2:00 AM.
        xyz.setStartRule(Calendar.JULY, 1, Calendar.SUNDAY, 2);

        // Change the ending time for daylight savings time to
        // the last Sunday in July at 2:00 AM.
        xyz.setEndRule(Calendar.JULY, -1, Calendar.SUNDAY, 2);

        // Determine if a few dates are in daylight savings time.
        Date d1 = new Date(105, Calendar.JULY, 3, 01, 00, 00);
        Date d2 = new Date(105, Calendar.JULY, 3, 03, 00, 00);
        Date d3 = new Date(105, Calendar.DECEMBER, 31, 00, 00, 00);
```

```
        boolean isD1InDaylight = xyz.inDaylightTime(d1);
        boolean isD2InDaylight = xyz.inDaylightTime(d2);
        boolean isD3InDaylight = xyz.inDaylightTime(d3);

        System.out.println("d1 is in DST? " + isD1InDaylight);
        System.out.println("d2 is in DST? " + isD2InDaylight);
        System.out.println("d3 is in DST? " + isD3InDaylight);
    }
}

/*
Output:
d1 is in DST? false
d2 is in DST? true
d3 is in DST? false
*/
```

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone.setStartYear Method

Sets the year when the rules governing this time zone begin.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setStartYear(
    int startYear);
```

Parameters

startYear

The year when the rules governing this time zone begin. This value is the current year - 1900.

Example

The following example changes the daylight savings time rules for a made-up time zone starting in the year 2005.

```
// simpletimezone_setstartyear.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create our own time zone named XYZ with an offset of
        // 100 milliseconds from GMT.  Set the daylight savings
        // time period from the first Sunday in April to the last
        // Sunday in October.
        SimpleTimeZone xyz = new SimpleTimeZone(100, "XYZ",
            Calendar.APRIL, 1, Calendar.SUNDAY, 2,
            Calendar.OCTOBER, -1, Calendar.SUNDAY, 2);
        TimeZone.setDefault(xyz);

        // Change the daylight savings time rules effective in the
        // year 2005.
        xyz.setStartYear(105);

        // Change the starting time for daylight savings time to
        // the first Sunday in July at 2:00 AM.
        xyz.setStartRule(Calendar.JULY, 1, Calendar.SUNDAY, 2);

        // Change the ending time for daylight savings time to
        // the last Sunday in July at 2:00 AM.
        xyz.setEndRule(Calendar.JULY, -1, Calendar.SUNDAY, 2);

        // Determine if a few dates are in daylight savings time.
        Date d1 = new Date(105, Calendar.APRIL, 15, 00, 00, 00);
        Date d2 = new Date(105, Calendar.JULY, 15, 00, 00, 00);
        Date d3 = new Date(105, Calendar.DECEMBER, 31, 00, 00, 00);

        boolean isD1InDaylight = xyz.inDaylightTime(d1);
        boolean isD2InDaylight = xyz.inDaylightTime(d2);
        boolean isD3InDaylight = xyz.inDaylightTime(d3);

        System.out.println("d1 is in DST? " + isD1InDaylight);
        System.out.println("d2 is in DST? " + isD2InDaylight);
        System.out.println("d3 is in DST? " + isD3InDaylight);
    }
}

/*
```

```
Output:
d1 is in DST? false
d2 is in DST? true
d3 is in DST? false
*/
```

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SimpleTimeZone.useDaylightTime Method

Determines whether the current time zone uses daylight savings time.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean useDaylightTime();
```

Return Value

true if the current time zone uses daylight savings time; false otherwise.

Example

The following example determines whether a given time zone uses daylight savings time.

```
// simpletimezone_usedaylighttime.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone tz = TimeZone.getDefault();
        String name = tz.getID();
        System.out.println(name + " uses DST? " +
            tz.useDaylightTime());
    }
}

/*
Output:
PST uses DST? true
*/
```

See Also

Reference

SimpleTimeZone Class

Concepts

SimpleTimeZone Members

java.util Package

SortedMap Interface

Represents a specialization of a Map object containing elements sorted by their key.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.SortedMap
    extends java.util.Map
```

Example

The following example demonstrates the firstKey, lastKey, headMap, and tailMap methods of the SortedMap class.

```
// sortedmap_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("3", "Sally Abolrous");
        tm.put("2", "Craig Combel");
        tm.put("5", "Pille Mandla");

        // Display the lowest key:
        System.out.println("The lowest key value is: " +
            tm.firstKey());

        // Display the highest key:
        System.out.println("The highest key value is: " +
            tm.lastKey());

        // Display the headMap:
        System.out.println("The head map is:\n" + tm.headMap("5"));

        // Display the tailMap:
        System.out.println("The tail map is:\n" + tm.tailMap("3"));
    }
}

/*
Output:
The lowest key value is: 2
The highest key value is: 5
The head map is:
{2=Craig Combel, 3=Sally Abolrous}
The tail map is:
{3=Sally Abolrous, 5=Pille Mandla}
*/
```

See Also

Concepts

SortedMap Members

java.util Package

SortedMap Members

Represents a specialization of a Map object containing elements sorted by their key.

The following tables list the members exposed by the SortedMap type.

Public Methods

Name	Description
comparator	Gets the Comparator object that is used to sort the elements in the map.
firstKey	Returns the first key in the map.
headMap	Returns a SortedMap object containing all the elements in this sorted map from the first element up to but not including the specified element.
lastKey	Returns the last key in the map.
subMap	Returns a SortedMap object containing all the elements in this sorted map with keys between the two specified elements.
tailMap	Returns a SortedMap object containing all the elements in this sorted map from the specified element to the last element.

See Also

Reference

SortedMap Interface

Concepts

java.util Package

SortedMap Methods

Public Methods

Name	Description
comparator	Gets the Comparator object that is used to sort the elements in the map.
firstKey	Returns the first key in the map.
headMap	Returns a SortedMap object containing all the elements in this sorted map from the first element up to but not including the specified element.
lastKey	Returns the last key in the map.
subMap	Returns a SortedMap object containing all the elements in this sorted map with keys between the two specified elements.
tailMap	Returns a SortedMap object containing all the elements in this sorted map from the specified element to the last element.

See Also

Reference

SortedMap Interface

Concepts

java.util Package

SortedMap.comparator Method

Gets the Comparator object that is used to sort the elements in the map.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Comparator comparator();
```

Return Value

The Comparator object that is used to sort the elements in the map.

See Also

Reference

SortedMap Interface

Concepts

SortedMap Members

java.util Package

SortedMap.firstKey Method

Returns the first key in the map.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object firstKey();
```

Return Value

The first key in the map.

Example

The following example displays the first key in the SortedMap object.

```
// sortedmap_firstkey.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("3", "Sally Abolrous");
        tm.put("2", "Craig Combel");
        tm.put("5", "Pille Mandla");

        // Display the lowest key:
        System.out.println("The lowest key value is: " +
            tm.firstKey());
    }
}

/*
Output:
The lowest key value is: 2
*/
```

See Also

Reference

SortedMap Interface

Concepts

SortedMap Members

java.util Package

SortedMap.headMap Method

Returns a SortedMap object containing all the elements in this sorted map from the first element up to but not including the specified element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.SortedMap headMap(
    java.lang.Object toV);
```

Parameters

toV

The element to compare with the elements in the SortedMap object. All elements with keys less than this element will be inserted into the returned map.

Return Value

A new instance of a SortedMap object containing all the elements in this sorted map with keys less than the given parameter toV.

Example

The following example displays a map of all elements in the SortedMap less than "5".

```
// sortedmap_headmap.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1", "Sally Abolrous");
        tm.put("3", "Craig Combel");
        tm.put("5", "Pille Mandla");

        // Display the headMap:
        System.out.println("The head map is:\n" + tm.headMap("5"));
    }
}

/*
Output:
The head map is:
{1=Sally Abolrous, 3=Craig Combel}
*/
```

See Also

Reference

SortedMap Interface

Concepts

SortedMap Members

java.util Package

SortedMap.lastKey Method

Returns the last key in the map.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object lastKey();
```

Return Value

The last key in the map.

Example

The following example displays the last key in the SortedMap object.

```
// sortedmap_lastkey.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("3", "Sally Abolrous");
        tm.put("2", "Craig Combel");
        tm.put("5", "Pille Mandla");

        // Display the highest key:
        System.out.println("The highest key value is: " +
            tm.lastKey());
    }
}

/*
Output:
The highest key value is: 5
*/
```

See Also

Reference

SortedMap Interface

Concepts

SortedMap Members

java.util Package

SortedMap.subMap Method

Returns a SortedMap object containing all the elements in this sorted map with keys between the two specified elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.SortedMap subMap(
    java.lang.Object fromV,
    java.lang.Object toV);
```

Parameters

fromV

The element representing the first element to be inserted into the returned SortedMap object. All elements with keys greater than or equal to this element and less than or equal to toV will be inserted into the returned map.

toV

The element to compare with the elements in the SortedMap object. All elements with keys less than this element and greater than or equal to fromV will be inserted into the returned map.

Return Value

A new instance of a SortedMap object containing all the elements in this sorted map with keys greater than or equal to the given parameter fromV and less than the given parameter toV.

Example

The following example displays a map of all elements in the SortedMap in the range of [1, 3).

```
// sortedmap_submap.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1", "Sally Abolrous");
        tm.put("2", "Craig Combel");
        tm.put("3", "Pille Mandla");

        // Display the SortedMap:
        System.out.println(tm.subMap("1", "3"));
    }
}

/*
Output:
{1=Sally Abolrous, 2=Craig Combel}
*/
```

See Also

Reference

SortedMap Interface

Concepts

SortedMap Members

java.util Package

SortedMap.tailMap Method

Returns a SortedMap object containing all the elements in this sorted map from the specified element to the last element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.SortedMap tailMap(
    java.lang.Object fromV);
```

Parameters

fromV

The element representing the first element to be inserted into the returned SortedMap object. All elements with keys greater than or equal to this element will be inserted into the returned map.

Return Value

A new instance of a SortedMap object containing all the elements in this sorted map with keys greater than or equal to the given parameter fromV.

Example

The following example displays a map of all elements in the SortedMap greater than or equal to "2".

```
// sortedmap_tailmap.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1", "Sally Abolrous");
        tm.put("2", "Craig Combel");
        tm.put("3", "Pille Mandla");

        // Display the tailMap:
        System.out.println("The tail map is:\n" + tm.tailMap("2"));
    }
}

/*
Output:
The tail map is:
{2=Craig Combel, 3=Pille Mandla}
*/
```

See Also

Reference

SortedMap Interface

Concepts

SortedMap Members

java.util Package

SortedSet Interface

Represents a specialization of a Set object containing sorted elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public interface java.util.SortedSet
    extends java.util.Set
```

Example

The following example demonstrates the first, last, headSet, and tailSet methods of the SortedSet interface.

```
// sortedset_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Add some elements:
        ts.add(new Integer(5));
        ts.add(new Integer(4));
        ts.add(new Integer(9));

        // Display the first and the last:
        System.out.println("The first element: " + ts.first());
        System.out.println("The last element: " + ts.last());

        // Display the headSet and the tailSet:
        System.out.println("The headSet: " + ts.headSet(ts.last()));
        System.out.println("The tailSet: " + ts.tailSet(ts.last()));
    }
}

/*
Output:
The first element: 4
The last element: 9
The headSet: [4, 5]
The tailSet: [9]
*/
```

See Also

Concepts

SortedSet Members

java.util Package

SortedSet Members

Represents a specialization of a Set object containing sorted elements.

The following tables list the members exposed by the SortedSet type.

Public Methods

Name	Description
comparator	Gets the Comparator object that is used to sort the elements in the set.
first	Returns the first element in the set.
headSet	Returns a SortedSet object containing all the elements in this sorted set from the first element up to but not including the specified element.
last	Returns the last element in the set.
subSet	Returns a SortedSet object containing all the elements in this sorted set between the two specified elements.
tailSet	Returns a SortedSet object containing all the elements in this sorted set from the specified element to the last element.

See Also

Reference

SortedSet Interface

Concepts

java.util Package

SortedSet Methods

Public Methods

Name	Description
comparator	Gets the Comparator object that is used to sort the elements in the set.
first	Returns the first element in the set.
headSet	Returns a SortedSet object containing all the elements in this sorted set from the first element up to but not including the specified element.
last	Returns the last element in the set.
subSet	Returns a SortedSet object containing all the elements in this sorted set between the two specified elements.
tailSet	Returns a SortedSet object containing all the elements in this sorted set from the specified element to the last element.

See Also

Reference

SortedSet Interface

Concepts

java.util Package

SortedSet.comparator Method

Gets the Comparator object that is used to sort the elements in the set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.Comparator comparator();
```

Return Value

The Comparator object that is used to sort the elements in the set.

See Also

Reference

SortedSet Interface

Concepts

SortedSet Members

java.util Package

SortedSet.first Method

Returns the first element in the set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object first();
```

Return Value

The first element in the set.

Example

The following example displays the first element in a SortedSet object.

```
// sortedset_first.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Add some elements:
        ts.add(new Integer(5));
        ts.add(new Integer(4));
        ts.add(new Integer(9));

        // Display the first and the last:
        System.out.println("The first element: " + ts.first());
        System.out.println("The last element: " + ts.last());
    }
}

/*
Output:
The first element: 4
The last element: 9
*/
```

See Also

Reference

SortedSet Interface

Concepts

SortedSet Members

java.util Package

SortedSet.headSet Method

Returns a SortedSet object containing all the elements in this sorted set from the first element up to but not including the specified element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.SortedSet headSet(
    java.lang.Object toV);
```

Parameters

toV

The element to compare with the elements in the SortedSet object. All elements less than this element will be inserted into the returned set.

Return Value

A new instance of a SortedSet object containing all the elements in this sorted set less than the given parameter toV.

Example

The following example demonstrates how to create a set containing all elements in the set up to but not including the last element.

```
// sortedset_headset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Add some elements:
        ts.add(new Integer(5));
        ts.add(new Integer(4));
        ts.add(new Integer(9));

        // Display the headSet and the tailSet:
        System.out.println("The headSet: " + ts.headSet(ts.last()));
        System.out.println("The tailSet: " + ts.tailSet(ts.last()));
    }
}

/*
Output:
The headSet: [4, 5]
The tailSet: [9]
*/
```

See Also

Reference

SortedSet Interface

Concepts

SortedSet Members

java.util Package

SortedSet.last Method

Returns the last element in the set.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.lang.Object last();
```

Return Value

The last element in the set.

Example

The following example displays the last element in a SortedSet object.

```
// sortedset_last.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Add some elements:
        ts.add(new Integer(5));
        ts.add(new Integer(4));
        ts.add(new Integer(9));

        // Display the first and the last:
        System.out.println("The first element: " + ts.first());
        System.out.println("The last element: " + ts.last());
    }
}

/*
Output:
The first element: 4
The last element: 9
*/
```

See Also

Reference

SortedSet Interface

Concepts

SortedSet Members

java.util Package

SortedSet.subSet Method

Returns a SortedSet object containing all the elements in this sorted set between the two specified elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.SortedSet subSet(
    java.lang.Object fronV,
    java.lang.Object toV);
```

Parameters

fronV

The element representing the first element to be inserted into the returned SortedSet object. All elements greater than or equal to this element and less than or equal to toV will be inserted into the returned set.

toV

The element to compare with the elements in the SortedSet object. All elements less than this element and greater than or equal to fromV will be inserted into the returned set.

Return Value

A new instance of a SortedSet object containing all the elements in this sorted set greater than or equal to the given parameter fromV and less than the given parameter toV.

Example

The following example demonstrates how to create a sub set of a SortedSet object.

```
// sortedset_subset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Declare some Double objects:
        Double x = new Double(5.1);
        Double y = new Double(7.3);
        Double z = new Double(9.2);

        // Add some elements:
        ts.add(x);
        ts.add(y);
        ts.add(z);

        System.out.println("The TreeSet elements are: " +
                           "x=" + x + ", y=" + y + ", z=" + z);

        // Display the subsets between two different elements:
        System.out.println("The subset from x to z is: " +
                            ts.subSet(x, z));
        System.out.println("The subset from x to y is: " +
                            ts.subSet(x, y));
    }
}

/*
```

```
Output:
The TreeSet elements are: x=5.1, y=7.3, z=9.2
The subset from x to z is: [5.1, 7.3]
The subset from x to y is: [5.1]
*/
```

See Also

Reference

SortedSet Interface

Concepts

SortedSet Members

java.util Package

SortedSet.tailSet Method

Returns a SortedSet object containing all the elements in this sorted set from the specified element to the last element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract java.util.SortedSet tailSet(
    java.lang.Object fromV);
```

Parameters

fromV

The element representing the first element to be inserted into the returned SortedSet object. All elements greater than or equal to this element will be inserted into the returned set.

Return Value

A new instance of a SortedSet object containing all the elements in this sorted set greater than or equal to the given parameter fromV.

Example

The following example demonstrates how to create a set containing all elements from the specified element until the end of the set.

```
// sortedset_tailset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Add some elements:
        ts.add(new Integer(5));
        ts.add(new Integer(4));
        ts.add(new Integer(9));

        // Display the headSet and the tailSet:
        System.out.println("The headSet: " + ts.headSet(ts.last()));
        System.out.println("The tailSet: " + ts.tailSet(ts.last()));
    }
}

/*
Output:
The headSet: [4, 5]
The tailSet: [9]
*/
```

See Also

Reference

SortedSet Interface

Concepts

SortedSet Members

java.util Package

Stack Class (J#)

Represents a collection of objects that are accessed in a last-in, first out fashion. Elements are pushed onto the top of the stack, and the last element pushed on is the first element to be popped off.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.Stack
    extends java.util.Vector
```

Example

The following example demonstrates the empty, peek, peekTop, pop, push, and search methods of the Stack class.

```
// stack_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Stack and populate it with a few elements.
        Stack stk = new Stack();
        stk.push("A");
        stk.push("B");
        stk.push("C");

        // Determine if the stack is empty.
        if (!stk.empty())
        {
            // Take a look at the top element.
            System.out.println("The top element on the Stack is " +
                stk.peek());
        }

        // Pop off the top element.
        Object top = stk.pop();
        System.out.println("The element " + top.toString() +
            " was removed from the Stack.  The new top element is " +
            stk.peekTop());

        // Search for a few elements.
        String search1 = new String("A");
        String search2 = new String("C");

        int search1Pos = stk.search(search1);
        if (search1Pos > 0)
        {
            System.out.println(search1 + " is at position " +
                search1Pos + " in the Stack.");
        }
        else
        {
            System.out.println(search1 + " is not on the Stack.");
        }

        int search2Pos = stk.search(search2);
        if (search2Pos > 0)
        {
            System.out.println(search2 + " is at position " +
                search2Pos + " in the Stack.");
```

```
            }
            else
            {
                System.out.println(search2 + " is not on the Stack.");
            }
        }
    }

    /*
    Output:
    The top element on the Stack is C
    The element C was removed from the Stack.  The new top element is B
    A is at position 2 in the Stack.
    C is not on the Stack.
    */
```

Inheritance Hierarchy

java.lang.Object

java.util.AbstractCollection

java.util.AbstractList

java.util.Vector

java.util.Stack

See Also

Concepts

Stack Members

java.util Package

Stack Members (J#)

Represents a collection of objects that are accessed in a last-in, first out fashion. Elements are pushed onto the top of the stack, and the last element pushed on is the first element to be popped off.

The following tables list the members exposed by the Stack type.

Public Constructors

Name	Description
Stack	Initializes a new instance of a Stack object.

Public Fields

Name	Description
capacityIncrement	A value representing the size to increase the vector once its capacity is reached.(inherited from Vector)
elementCount	The number of elements currently stored in the vector.(inherited from Vector)
elementData	An internal static array containing the elements of the vector.(inherited from Vector)
modCount	(inherited from AbstractList)

Public Methods

Name	Description
add	Overloaded. Adds an element to the vector. (inherited from Vector)
addAll	Overloaded. Adds multiple elements to the vector at one time. (inherited from Vector)
addElement	Adds an element to the end of the vector. The vector is automatically grown if there is insufficient room to add the element. (inherited from Vector)
capacity	Returns the number of elements capable of being stored in the vector. (inherited from Vector)
clear	Removes all elements from the vector. (inherited from Vector)
contains	Determines whether the vector contains a given element. (inherited from Vector)
containsAll	Determines whether all the elements in the given collection are contained in the vector. (inherited from Vector)
copyInto	Copies all the elements in the vector into an array. (inherited from Vector)
elementAt	Returns the element at the specified index. The element is not removed from the vector. (inherited from Vector)
elements	Returns an Enumeration object that can be used to efficiently traverse the elements in the vector. (inherited from Vector)
empty	Determines whether the stack is empty or if it contains elements.
ensureCapacity	Determines whether the vector can hold the given number of elements. If it cannot, a new vector is created that is large enough to store the desired number of elements and the contents of the old vector are copied into the newly created vector. (inherited from Vector)

equals	Determines whether two vectors are equal. Two vectors are considered equal if they contain exactly the same elements. (inherited from Vector)
firstElement	Returns the first element in the vector. The element is not removed from the vector. (inherited from Vector)
get	Returns the element at a given index. The element is not removed from the vector. (inherited from Vector)
hashCode	Provides a hash value representing this vector. (inherited from Vector)
getClass	(inherited from Object)
indexOf	Overloaded. Returns the index of the first occurrence of a given element. (inherited from Vector)
insertElementAt	Inserts an element into the vector at the given index. The vector is automatically grown if there is insufficient room to add the element. (inherited from Vector)
isEmpty	Determines whether the vector contains any elements. (inherited from Vector)
iterator	(inherited from AbstractList)
lastElement	Returns the last element in the vector. The element is not removed from the vector. (inherited from Vector)
lastIndexOf	Overloaded. Returns the index of the last occurrence of a given element. (inherited from Vector)
listIterator	Overloaded. (inherited from AbstractList)
clone	Creates a new instance of a Vector object that is a shallow copy of the vector. (inherited from Vector)
peek	Returns the top element on the stack but does not remove it from the stack.
peekTop	Returns the top element on the stack but does not remove it from the stack.
pop	Removes and returns the top element from the stack.
push	Adds an element to the top of the stack.
remove	Overloaded. Removes an element from the vector. (inherited from Vector)
removeAll	Removes all the elements contained in a collection from the vector. If an element appears more than once in the vector, all instances are removed. The elements in the vector beyond these elements are then shifted down. (inherited from Vector)
removeAllElements	Removes all elements from the vector. (inherited from Vector)
removeElement	Removes the first occurrence of an element from the vector. The elements in the vector beyond this element are then shifted down. (inherited from Vector)
removeElementAt	Removes the element from the vector at the specified index. The elements in the vector beyond this index are then shifted down. (inherited from Vector)
removeRange	Removes all the elements from the vector starting with the element at fromIx to the element at one less than toIx. (inherited from Vector)

retainAll	Removes all elements from the vector that are not contained in the given collection. (inherited from Vector)
search	Searches for an element on a stack.
set	Sets the element at the given index to the given element. (inherited from Vector)
setElementAt	Sets the element at the given index to the given element. (inherited from Vector)
setSize	Sets the size of the vector to the given value. If the given size is greater than the current size of the vector, t hen the contents of the old vector are copied into a new vector of the given size. If the given size is less tha n the current size of the vector, then some elements will be removed from the vector. (inherited from Vector)
size	Returns the number of elements in the vector. (inherited from Vector)
subList	Returns a List of all the elements in the vector starting with the element at fromIx to the element at one les s than tolx. (inherited from Vector)
toArray	Overloaded. Converts a vector into a static array. (inherited from Vector)
toString	Displays a human-readable representation of a vector. (inherited from Vector)
trimToSize	Changes the capacity of a vector to match its size. (inherited from Vector)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Stack Class

Concepts

java.util Package

Stack Fields

Public Fields

Name	Description
capacityIncrement	A value representing the size to increase the vector once its capacity is reached. (inherited from Vector)
elementCount	The number of elements currently stored in the vector. (inherited from Vector)
elementData	An internal static array containing the elements of the vector. (inherited from Vector)
modCount	(inherited from AbstractList)

See Also

Reference

Stack Class

Concepts

java.util Package

Stack Constructor

Initializes a new instance of a Stack object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Stack();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Stack Class

Concepts

Stack Members

java.util Package

Stack Methods

Public Methods

Name	Description
add	Overloaded. Adds an element to the vector. (inherited from Vector)
addAll	Overloaded. Adds multiple elements to the vector at one time. (inherited from Vector)
addElement	Adds an element to the end of the vector. The vector is automatically grown if there is insufficient room to add the element. (inherited from Vector)
capacity	Returns the number of elements capable of being stored in the vector. (inherited from Vector)
clear	Removes all elements from the vector. (inherited from Vector)
contains	Determines whether the vector contains a given element. (inherited from Vector)
containsAll	Determines whether all the elements in the given collection are contained in the vector. (inherited from Vector)
copyInto	Copies all the elements in the vector into an array. (inherited from Vector)
elementAt	Returns the element at the specified index. The element is not removed from the vector. (inherited from Vector)
elements	Returns an Enumeration object that can be used to efficiently traverse the elements in the vector. (inherited from Vector)
empty	Determines whether the stack is empty or if it contains elements.
ensureCapacity	Determines whether the vector can hold the given number of elements. If it cannot, a new vector is created that is large enough to store the desired number of elements and the contents of the old vector are copied into the newly created vector. (inherited from Vector)
equals	Determines whether two vectors are equal. Two vectors are considered equal if they contain exactly the same elements. (inherited from Vector)
firstElement	Returns the first element in the vector. The element is not removed from the vector. (inherited from Vector)
get	Returns the element at a given index. The element is not removed from the vector. (inherited from Vector)
hashCode	Provides a hash value representing this vector. (inherited from Vector)
getClass	(inherited from Object)
indexOf	Overloaded. Returns the index of the first occurrence of a given element. (inherited from Vector)
insertElementAt	Inserts an element into the vector at the given index. The vector is automatically grown if there is insufficient room to add the element. (inherited from Vector)
isEmpty	Determines whether the vector contains any elements. (inherited from Vector)

iterator	(inherited from AbstractList)
lastElement	Returns the last element in the vector. The element is not removed from the vector. (inherited from Vector)
lastIndexOf	Overloaded. Returns the index of the last occurrence of a given element. (inherited from Vector)
listIterator	Overloaded. (inherited from AbstractList)
clone	Creates a new instance of a Vector object that is a shallow copy of the vector. (inherited from Vector)
peek	Returns the top element on the stack but does not remove it from the stack.
peekTop	Returns the top element on the stack but does not remove it from the stack.
pop	Removes and returns the top element from the stack.
push	Adds an element to the top of the stack.
remove	Overloaded. Removes an element from the vector. (inherited from Vector)
removeAll	Removes all the elements contained in a collection from the vector. If an element appears more than once in the vector, all instances are removed. The elements in the vector beyond these elements are then shifted down. (inherited from Vector)
removeAllElements	Removes all elements from the vector. (inherited from Vector)
removeElement	Removes the first occurrence of an element from the vector. The elements in the vector beyond this element are then shifted down. (inherited from Vector)
removeElementAt	Removes the element from the vector at the specified index. The elements in the vector beyond this index are then shifted down. (inherited from Vector)
removeRange	Removes all the elements from the vector starting with the element at fromIx to the element at one less than toIx. (inherited from Vector)
retainAll	Removes all elements from the vector that are not contained in the given collection. (inherited from Vector)
search	Searches for an element on a stack.
set	Sets the element at the given index to the given element. (inherited from Vector)
setElementAt	Sets the element at the given index to the given element. (inherited from Vector)
setSize	Sets the size of the vector to the given value. If the given size is greater than the current size of the vector, then the contents of the old vector are copied into a new vector of the given size. If the given size is less than the current size of the vector, then some elements will be removed from the vector. (inherited from Vector)
size	Returns the number of elements in the vector. (inherited from Vector)
subList	Returns a List of all the elements in the vector starting with the element at fromIx to the element at one less than toIx. (inherited from Vector)

toArray	Overloaded. Converts a vector into a static array. (inherited from Vector)
toString	Displays a human-readable representation of a vector. (inherited from Vector)
trimToSize	Changes the capacity of a vector to match its size. (inherited from Vector)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Stack Class

Concepts

java.util Package

Stack.empty Method

Determines whether the stack is empty or if it contains elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean empty();
```

Return Value

true if the stack is empty; false otherwise.

Example

The following example demonstrates how to determine if a Stack is empty.

```
// stack_empty.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Stack and populate it with a few elements.
        Stack stk = new Stack();
        stk.push(new String("A"));
        stk.push(new String("B"));
        stk.push(new String("C"));

        if (!stk.empty())
        {
            // Do something useful with the Stack here.
        }
    }
}
```

See Also

Reference

Stack Class

Concepts

Stack Members

java.util Package

Stack.peek Method

Returns the top element on the stack but does not remove it from the stack.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.Object peek();
```

Return Value

The top element on the stack, which is the last element that was added to the stack.

Example

The following example shows the top element on the Stack.

```
// stack_peek.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Stack and populate it with a few elements.
        Stack stk = new Stack();
        stk.push(new String("A"));
        stk.push(new String("B"));
        stk.push(new String("C"));

        // Look at the top element on the Stack.
        Object o = stk.peek();
        System.out.println("peek: " + o.toString());
    }
}

/*
Output:
peek: C
*/
```

See Also

Reference

Stack Class

Concepts

Stack Members

java.util Package

Stack.peekTop Method

Returns the top element on the stack but does not remove it from the stack.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object peekTop();
```

Return Value

The top element on the stack, which is the last element that was added to the stack.

Example

The following example shows the top element on the Stack.

```
// stack_peektop.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Stack and populate it with a few elements.
        Stack stk = new Stack();
        stk.push(new String("A"));
        stk.push(new String("B"));
        stk.push(new String("C"));

        // Look at the top element on the Stack.
        Object o = stk.peekTop();
        System.out.println("peekTop: " + o.toString());
    }
}

/*
Output:
peekTop: C
*/
```

See Also

Reference

Stack Class

Concepts

Stack Members

java.util Package

Stack.pop Method

Removes and returns the top element from the stack.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.Object pop();
```

Return Value

The top element on the stack, which is the last element that was added to the stack.

Example

The following example removes the top element from the Stack.

```
// stack_pop.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Stack and populate it with a few elements.
        Stack stk = new Stack();
        stk.push(new String("A"));
        stk.push(new String("B"));
        stk.push(new String("C"));

        // Look at and remove the top element on the Stack.
        Object o = stk.pop();
        System.out.println("pop: " + o.toString());
        System.out.println("new top: " + stk.peek().toString());
    }
}

/*
Output:
pop: C
new top: B
*/
```

See Also

Reference

Stack Class

Concepts

Stack Members

java.util Package

Stack.push Method

Adds an element to the top of the stack.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object push(
    java.lang.Object item);
```

Parameters

item

The element to be added to the top of the stack.

Return Value

The element that was added to the top of the stack.

Example

The following example adds an element to the top of the Stack.

```
// stack_push.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Stack and populate it with a few elements.
        Stack stk = new Stack();
        stk.push(new String("A"));
        stk.push(new String("B"));
        stk.push(new String("C"));

        // Add another element to the top of the Stack.
        stk.push(new String("D"));
        System.out.println("new top: " + stk.peek().toString());
    }
}

/*
Output:
new top: D
*/
```

See Also

Reference

Stack Class

Concepts

Stack Members

java.util Package

Stack.search Method

Searches for an element on a stack.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized int search(
    java.lang.Object obj);
```

Parameters

obj

The element to search the stack for.

Return Value

The position of the element on the stack. The top of the stack is position 1, the element immediately below that is position 2, etc. If the element being searched for is not on the stack, then -1 is returned.

Example

The following example demonstrates how to search a Stack for a particular element. The position within the Stack is then displayed.

```
// stack_search.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Stack and populate it with a few elements.
        Stack stk = new Stack();
        stk.push(new String("A"));
        stk.push(new String("B"));
        stk.push(new String("C"));

        // Look for those elements as well as an element that doesn't
        // exist on the Stack.
        String strD = new String("D");
        String strC = new String("C");
        String strB = new String("B");
        String strA = new String("A");

        System.out.println("search D: " + stk.search(strD));
        System.out.println("search C: " + stk.search(strC));
        System.out.println("search B: " + stk.search(strB));
        System.out.println("search A: " + stk.search(strA));
    }
}

/*
Output:
search D: -1
search C: 1
search B: 2
search A: 3
*/
```

See Also

Reference

Stack Class
Concepts
Stack Members
java.util Package

StringTokenizer Class

Provides methods to parse a string and split it into tokens based on a delimiter.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.StringTokenizer
    extends java.lang.Object
    implements java.util.Enumeration
```

Example

The following example demonstrates the countTokens, hasMoreElements, hasMoreTokens, nextElement, and nextToken methods of the StringTokenizer class.

```
// stringtokenizer_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        String str =
            "This is/some text/that I am/parsing/using StringTokenizer/.";

        StringTokenizer strTokElem =
            new StringTokenizer(str, "/", true);
        System.out.println("Elements...");
        while (strTokElem.hasMoreElements())
        {
            System.out.println(strTokElem.nextElement());
        }

        StringTokenizer strTokToken =
            new StringTokenizer(str, "/", false);
        System.out.println("Tokens...");
        while (strTokToken.hasMoreTokens())
        {
            System.out.println(strTokToken.nextToken());
        }

        StringTokenizer strTok =
            new StringTokenizer(str, "/", false);
        System.out.println("Count...");
        System.out.println(strTok.countTokens());
    }
}

/*
Output:
Elements...
This is
/
some text
/
that I am
/
parsing
/
using StringTokenizer
/
```

```
.
Tokens...
This is
some text
that I am
parsing
using StringTokenizer
.
Count...
6
*/
```

Inheritance Hierarchy

java.lang.Object

java.util.StringTokenizer

See Also

Concepts

StringTokenizer Members

java.util Package

StringTokenizer Members

Provides methods to parse a string and split it into tokens based on a delimiter.

The following tables list the members exposed by the StringTokenizer type.

Public Constructors

Name	Description
StringTokenizer	Overloaded. Initializes a new instance of a StringTokenizer object.

Public Methods

Name	Description
countTokens	Counts the number of remaining tokens in the string. Tokens that have already been parsed are not included in the total.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
hasMoreElements	Determines whether there are more elements in the string to be enumerated. For StringTokenizer objects, an element is a token.
hasMoreTokens	Determines whether there are more tokens in the string to be parsed.
clone	Creates a new instance of a StringTokenizer object that is a shallow copy of the existing StringTokenizer object.
nextElement	Gets the next element in the string during enumeration. For StringTokenizer objects, an element is a token.
nextToken	Overloaded. Gets the next token in the string during parsing.
toString	Overridden. Displays a human-readable representation of a StringTokenizer object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringTokenizer Class

Concepts

java.util Package

StringTokenizer Constructor

Initializes a new instance of a StringTokenizer object.

Overload List

Name	Description
StringTokenizer (String)	Initializes a new instance of a StringTokenizer object using the given string.
StringTokenizer (String, String)	Initializes a new instance of a StringTokenizer object using the given string and delimiter.
StringTokenizer (String, String, boolean)	Initializes a new instance of a StringTokenizer object using the given string and delimiter. You can also specify whether the delimiter is to be retained and counted as a token.

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

StringTokenizer Constructor (String)

Initializes a new instance of a StringTokenizer object using the given string.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.StringTokenizer(
    java.lang.String str);
```

Parameters

str

The string to be tokenized.

See Also
Reference
StringTokenizer Class
Concepts
StringTokenizer Members
java.util Package

StringTokenizer Constructor (String, String)

Initializes a new instance of a StringTokenizer object using the given string and delimiter.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.StringTokenizer(
    java.lang.String str,
    java.lang.String delim);
```

Parameters

str

The string to be tokenized.

delim

The delimiter used to tokenize the string.

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

StringTokenizer Constructor (String, String, Boolean)

Initializes a new instance of a StringTokenizer object using the given string and delimiter. You can also specify whether the delimiter is to be retained and counted as a token.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.StringTokenizer(
    java.lang.String str,
    java.lang.String delim,
    boolean retDelim);
```

Parameters

str

The string to be tokenized.

delim

The delimiter used to tokenize the string.

retDelim

Indicates whether the delimiter is to be retained and counted as a token.

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

StringTokenizer Methods

Public Methods

Name	Description
countTokens	Counts the number of remaining tokens in the string. Tokens that have already been parsed are not included in the total.
equals	(inherited from Object)
hashCode	(inherited from Object)
getClass	(inherited from Object)
hasMoreElements	Determines whether there are more elements in the string to be enumerated. For StringTokenizer objects, an element is a token.
hasMoreTokens	Determines whether there are more tokens in the string to be parsed.
clone	Creates a new instance of a StringTokenizer object that is a shallow copy of the existing StringTokenizer object.
nextElement	Gets the next element in the string during enumeration. For StringTokenizer objects, an element is a token.
nextToken	Overloaded. Gets the next token in the string during parsing.
toString	Overridden. Displays a human-readable representation of a StringTokenizer object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

StringTokenizer Class

Concepts

java.util Package

StringTokenizer.countTokens Method

Counts the number of remaining tokens in the string. Tokens that have already been parsed are not included in the total.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int countTokens();
```

Return Value

The number of tokens in the string yet to be parsed

Example

The following example counts the number of tokens in a string.

```
// stringtokenizer_counttokens.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        String str =
            "This is/some text/that I am/parsing/using StringTokenizer/.";

        StringTokenizer strTok =
            new StringTokenizer(str, "/", false);

        System.out.println("Count...");
        System.out.println(strTok.countTokens());
    }
}

/*
Output:
Count...
6
*/
```

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

StringTokenizer.hasMoreElements Method

Determines whether there are more elements in the string to be enumerated. For StringTokenizer objects, an element is a token.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean hasMoreElements();
```

Return Value

true if there are more elements in the string yet to be parsed; false otherwise.

Example

The following example shows how to determine if there are more elements remaining in the string being tokenized.

```
// stringtokenizer_hasmoreelements.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        String str =
            "This is/some text/that I am/parsing/using StringTokenizer/.";

        StringTokenizer strTokElem =
            new StringTokenizer(str, "/", true);

        System.out.println("Elements...");
        while (strTokElem.hasMoreElements())
        {
            System.out.println(strTokElem.nextElement());
        }
    }
}

/*
Output:
Elements...
This is
/
some text
/
that I am
/
parsing
/
using StringTokenizer
/
.
*/
```

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

StringTokenizer.hasMoreTokens Method

Determines whether there are more tokens in the string to be parsed.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean hasMoreTokens();
```

Return Value

true if there are more tokens in the string yet to be parsed; false otherwise.

Example

The following example shows how to determine if there are more tokens remaining in the string being tokenized.

```
// stringtokenizer_hasmoretokens.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        String str =
            "This is/some text/that I am/parsing/using StringTokenizer/.";

        StringTokenizer strTokToken =
            new StringTokenizer(str, "/", false);

        System.out.println("Tokens...");
        while (strTokToken.hasMoreTokens())
        {
            System.out.println(strTokToken.nextToken());
        }
    }
}

/*
Output:
Tokens...
This is
some text
that I am
parsing
using StringTokenizer
.
*/
```

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

StringTokenizer.nextElement Method

Gets the next element in the string during enumeration. For StringTokenizer objects, an element is a token.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object nextElement();
```

Return Value

The next token in the string.

Example

The following example shows how to get the next element from the string being tokenized.

```
// stringtokenizer_nextelement.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        String str =
            "This is/some text/that I am/parsing/using StringTokenizer/.";

        StringTokenizer strTokElem =
            new StringTokenizer(str, "/", true);

        System.out.println("Elements...");
        while (strTokElem.hasMoreElements())
        {
            System.out.println(strTokElem.nextElement());
        }
    }
}

/*
Output:
Elements...
This is
/
some text
/
that I am
/
parsing
/
using StringTokenizer
/
.
*/
```

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

StringTokenizer.nextToken Method

Gets the next token in the string during parsing.

Overload List

Name	Description
StringTokenizer.nextToken ()	Gets the next token in the string during parsing.
StringTokenizer.nextToken (String)	Gets the next token in the string as determined by the given delimiter.

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

StringTokenizer.nextToken Method ()

Gets the next token in the string during parsing.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String nextToken();
```

Return Value

The next token in the string.

Example

The following example shows how to get the next token from the string being tokenized.

```
// stringtokenizer_nexttoken.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        String str =
            "This is/some text/that I am/parsing/using StringTokenizer/.";

        StringTokenizer strTokToken =
            new StringTokenizer(str, "/", false);

        System.out.println("Tokens...");
        while (strTokToken.hasMoreTokens())
        {
            System.out.println(strTokToken.nextToken());
        }
    }
}

/*
Output:
Tokens...
This is
some text
that I am
parsing
using StringTokenizer
.
*/
```

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

StringTokenizer.nextToken Method (String)

Gets the next token in the string as determined by the given delimiter.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String nextToken(
    java.lang.String delim);
```

Parameters

delim

The delimiter used to tokenize the string.

Return Value

The next token in the string.

Example

The following example shows how to get the next token (delineated by the "/" character) from the string being tokenized.

```
// stringtokenizer_nexttoken_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        String str =
            "This is/some text/that I am/parsing/using StringTokenizer/.";

        StringTokenizer strTokToken =
            new StringTokenizer(str, "/", false);

        System.out.println("Tokens...");
        while (strTokToken.hasMoreTokens())
        {
            System.out.println(strTokToken.nextToken("/"));
        }
    }
}

/*
Output:
Tokens...
This is
some text
that I am
parsing
using StringTokenizer
.
*/
```

See Also

Reference

StringTokenizer Class

Concepts

StringTokenizer Members

java.util Package

TimeZone Class

Represents an abstract class containing methods useful for calculations involving time zones, such as offsets from UTC and determining whether daylight savings time is in effect.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract class java.util.TimeZone
    extends java.lang.Object
    implements java.io.Serializable, java.lang.Cloneable
```

Example

The following example demonstrates the getAvailableIDs, getDefault, getID, getRawOffset, getTimeZone, and inDaylightTime methods of the TimeZone class.

```
// simpletimezone_overview_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone tz = TimeZone.getDefault();

        // Print out the ID and raw offset of the default
        // time zone.
        System.out.println("default ID = " + tz.getID());
        System.out.println("raw offset = " + tz.getRawOffset());

        // Print out the rest of the IDs known to the system.
        String[] availTZs = TimeZone.getAvailableIDs();
        for (int i = 0; i < availTZs.length; i++)
        {
            System.out.println(availTZs[i]);
        }

        // Get the time zone known by the ID "GMT".
        TimeZone gmt = tz.getTimeZone("GMT");
        if (gmt != null)
        {
            System.out.println("GMT raw offset = " +
                gmt.getRawOffset());
        }

        // Determine if a given date is in daylight savings time.
        // Friday, July 15, 2005 at midnight.
        Date d2 = new Date(
            105,
            Calendar.JULY,
            15,
            00,
            00,
            00);
        boolean isInDaylight = tz.inDaylightTime(d2);
        System.out.println("July 15 in DST? " + isInDaylight);
    }
}

/*
Output:
```

```
default ID = PST
raw offset = -28800000
GMT
UTC
ECT
EET
ART
EAT
MET
NET
PLT
IST
BST
VST
CTT
JST
ACT
AET
SST
NST
MIT
HST
AST
PST
PNT
MST
CST
EST
IET
PRT
CNT
AGT
BET
CAT
GMT raw offset = 0
July 15 in DST? true
*/
```

Inheritance Hierarchy

java.lang.Object

java.util.TimeZone

java.util.SimpleTimeZone

See Also

Concepts

TimeZone Members

java.util Package

TimeZone Members

Represents an abstract class containing methods useful for calculations involving time zones, such as offsets from UTC and determining whether daylight savings time is in effect.

The following tables list the members exposed by the TimeZone type.

Public Constructors

Name	Description
TimeZone	Initializes a new instance of a TimeZone object.

Public Methods

Name	Description
equals	(inherited from Object)
getAvailableIDs	Overloaded. Gets an array of all the valid IDs for all time zones around the world.
getDefault	Gets the default time zone.
hashCode	(inherited from Object)
getID	Gets the ID of the current time zone.
getOffset	Gets the offset of the time zone from UTC of the given date, including any effects of daylight savings time.
getRawOffset	Gets the offset of the time zone from UTC, ignoring daylight savings time.
getTimeZone	Gets a TimeZone object corresponding to the ID provided.
getClass	(inherited from Object)
inDaylightTime	Determines whether daylight savings time is in effect on the given date.
clone	Creates a new instance of a TimeZone object that is a shallow copy of the current object.
setDefault	Sets the default time zone. This will override the default time zone determined by the current locale of the system.
setID	Sets the ID used to represent the current time zone.
setRawOffset	Sets the offset of the time zone from UTC, ignoring daylight savings time.
toString	Overridden. Displays a human-readable representation of a TimeZone object.
useDaylightTime	Determines whether the current time zone uses daylight savings time.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

TimeZone Class
Concepts
java.util Package

TimeZone Constructor

Initializes a new instance of a TimeZone object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.TimeZone();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone Methods

Public Methods

Name	Description
equals	(inherited from Object)
getAvailableIDs	Overloaded. Gets an array of all the valid IDs for all time zones around the world.
getDefault	Gets the default time zone.
hashCode	(inherited from Object)
getID	Gets the ID of the current time zone.
getOffset	Gets the offset of the time zone from UTC of the given date, including any effects of daylight savings time.
getRawOffset	Gets the offset of the time zone from UTC, ignoring daylight savings time.
getTimeZone	Gets a TimeZone object corresponding to the ID provided.
getClass	(inherited from Object)
inDaylightTime	Determines whether daylight savings time is in effect on the given date.
clone	Creates a new instance of a TimeZone object that is a shallow copy of the current object.
setDefault	Sets the default time zone. This will override the default time zone determined by the current locale of the sys tem.
setID	Sets the ID used to represent the current time zone.
setRawOffset	Sets the offset of the time zone from UTC, ignoring daylight savings time.
toString	Overridden. Displays a human-readable representation of a TimeZone object.
useDaylightTime	Determines whether the current time zone uses daylight savings time.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

TimeZone Class

Concepts

java.util Package

TimeZone.clone Method

Creates a new instance of a TimeZone object that is a deep copy of the current object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

A new instance of a TimeZone object that is a deep copy of the current object.

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.getAvailableIDs Method

Gets an array of all the valid IDs for all time zones around the world.

Overload List

Name	Description
TimeZone.getAvailableIDs ()	Gets an array of all the valid IDs for all time zones around the world.
TimeZone.getAvailableIDs (int)	Gets an array of all the valid IDs for all time zones around the world that have the given offset from UTC.

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.getAvailableIDs Method ()

Gets an array of all the valid IDs for all time zones around the world.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.lang.String[] getAvailableIDs();
```

Return Value

An array of all the valid IDs for all time zones around the world. Each time zone is represented by a symbol such as "UTC" or "PST".

Example

The following example shows how to get all available time zone IDs.

```
// simpletimezone_getavailableids.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        String[] availTZs = TimeZone.getAvailableIDs();
        for (int i = 0; i < availTZs.length; i++)
        {
            System.out.println(availTZs[i]);
        }
    }
}

/*
Output:
GMT
UTC
ECT
EET
ART
EAT
MET
NET
PLT
IST
BST
VST
CTT
JST
ACT
AET
SST
NST
MIT
HST
AST
PST
PNT
MST
CST
EST
IET
```

```
PRT
CNT
AGT
BET
CAT
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.getAvailableIDs Method (Int32)

Gets an array of all the valid IDs for all time zones around the world that have the given offset from UTC.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.lang.String[] getAvailableIDs(
    int rawOffset);
```

Parameters

rawOffset

The offset, represented in milliseconds, of the time zone from UTC, ignoring daylight savings time. Areas east of UTC have positive offsets, while areas west of UTC have negative offsets.

Return Value

An array of all the valid IDs for all time zones around the world that have the given offset from UTC. Each time zone is represented by a symbol such as "UTC" or "PST".

Example

The following example shows how to get all available time zone IDs with an offset of -28800000 milliseconds.

```
// simpletimezone_getavailableids_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        String[] availTZsOffset = TimeZone.getAvailableIDs(-28800000);
        for (int i = 0; i < availTZsOffset.length; i++)
        {
            System.out.println(availTZsOffset[i]);
        }
    }
}

/*
Output:
PST
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.getDefault Method

Gets the default time zone.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.util.TimeZone getDefault();
```

Return Value

The default time zone. This can either be the default time zone for the current locale of the system, or the value set in the setDefault method.

Example

The following example shows how to get the default time zone for the system.

```
// simpletimezone_getdefault.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone def = SimpleTimeZone.getDefault();
        String id = def.getID();
        System.out.println("default time zone: " + id);
    }
}

/*
Output:
default time zone: PST
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.getID Method

Gets the ID of the current time zone.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.String getID();
```

Return Value

The ID of the current time zone, such as "UTC" or "PST".

Example

The following example shows how to get the ID for the default time zone for the system.

```
// simpletimezone_getid.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone def = SimpleTimeZone.getDefault();
        String id = def.getID();
        System.out.println("default time zone: " + id);
    }
}

/*
Output:
default time zone: PST
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.getOffset Method

Gets the offset of the time zone from UTC of the given date, including any effects of daylight savings time.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int getOffset(
    int era,
    int year,
    int month,
    int day,
    int dayOfWeek,
    int milliSeconds);
```

Parameters

era

The era of the date being used to calculate the offset. For the Gregorian calendar system, the value will be one of BC or AD.

year

The year of the date being used to calculate the offset. This value is the current year - 1900.

month

The month of the date being used to calculate the offset. This value is one of the month constants in the Calendar class, such as JANUARY.

day

The day of the month for the date being used to calculate the offset.

dayOfWeek

The day of the week for the date being used to calculate the offset. This value is one of the day constants in the Calendar class, such as SUNDAY.

milliSeconds

The time, in milliseconds, for the date being used to calculate the offset.

Return Value

The offset, represented in milliseconds, of the time zone from UTC, including any effects of daylight savings time. Areas east of UTC will get a positive return value, while areas west of UTC will get a negative return value.

Example

The following example shows the offset for the default time zone for both January 1, 2005 and July 1, 2005. In areas where daylight savings time is observed, the two offsets will be different.

```
// simpletimezone_getoffset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone def = SimpleTimeZone.getDefault();

        // Saturday, January 1, 2005 at midnight.
        int offset1 = def.getOffset(
            GregorianCalendar.AD,
            105,
```

```
            Calendar.JANUARY,
            1,
            Calendar.SATURDAY,
            0);
        System.out.println("January 1 offset: " + offset1);

        // Friday, July 1, 2005 at midnight.
        int offset2 = def.getOffset(
            GregorianCalendar.AD,
            105,
            Calendar.JULY,
            1,
            Calendar.FRIDAY,
            0);
        System.out.println("July 1 offset: " + offset2);
    }
}

/*
Output:
January 1 offset: -28800000
July 1 offset: -25200000
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.getRawOffset Method

Gets the offset of the time zone from UTC, ignoring daylight savings time.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract int getRawOffset();
```

Return Value

The offset, represented in milliseconds, of the time zone from UTC, ignoring daylight savings time. Areas east of UTC will get a positive return value, while areas west of UTC will get a negative return value.

Example

The following example shows how to get the raw offset for the default time zone. Daylight savings time does not factor into the raw offset.

```
// simpletimezone_getrawoffset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone def = SimpleTimeZone.getDefault();

        int offset = def.getRawOffset();
        System.out.println("Raw offset: " + offset);
    }
}

/*
Output:
Raw offset: -28800000
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.getTimeZone Method

Gets a TimeZone object corresponding to the ID provided.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized java.util.TimeZone getTimeZone(
    java.lang.String ID);
```

Parameters

ID

The ID of the time zone to be retrieved.

Return Value

A TimeZone object corresponding to the ID provided.

Example

The following example demonstrates how to get a time zone based on its ID.

```
// simpletimezone_gettimezone.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone tz = TimeZone.getTimeZone("PST");
        if (tz != null)
        {
            System.out.println("Raw offset of PST: " +
                tz.getRawOffset());
        }
    }
}

/*
Output:
Raw offset of PST: -28800000
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.inDaylightTime Method

Determines whether daylight savings time is in effect on the given date.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean inDaylightTime(
    java.util.Date date);
```

Parameters

date

The Date object used to determine if daylight savings time is in effect.

Return Value

true if daylight savings time is in effect on the given date; false otherwise.

Example

The following example shows how to determine if a Date is in daylight savings time.

```
// simpletimezone_indaylighttime.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone def = SimpleTimeZone.getDefault();

        // Saturday, January 1, 2005 at midnight.
        Date d1 = new Date(
            105,
            Calendar.JANUARY,
            1,
            00,
            00,
            00);
        boolean isInDaylight = def.inDaylightTime(d1);
        System.out.println("January 1 in DST? " + isInDaylight);

        // Friday, July 1, 2005 at midnight.
        Date d2 = new Date(
            105,
            Calendar.JULY,
            1,
            00,
            00,
            00);
        isInDaylight = def.inDaylightTime(d2);
        System.out.println("July 1 in DST? " + isInDaylight);
    }
}

/*
Output:
January 1 in DST? false
July 1 in DST? true
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.setDefault Method

Sets the default time zone. This will override the default time zone determined by the current locale of the system.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public static synchronized void setDefault(
    java.util.TimeZone def);
```

Parameters

def

A TimeZone object representing the default time zone.

Example

The following example demonstrates how to change the default time zone for the system.

```
// simpletimezone_setdefault.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create our own time zone named XYZ with an offset of
        // 100 milliseconds from GMT.  Set the daylight savings
        // time period from the first Sunday in April to the last
        // Sunday in October.  Make this the default time zone.
        SimpleTimeZone xyz = new SimpleTimeZone(100, "XYZ",
            Calendar.APRIL, 1, Calendar.SUNDAY, 2,
            Calendar.OCTOBER, -1, Calendar.SUNDAY, 2);
        TimeZone.setDefault(xyz);

        System.out.println("default time zone = " +
            TimeZone.getDefault().getID());
    }
}

/*
Output:
default time zone = XYZ
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.setID Method

Sets the ID used to represent the current time zone.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void setID(
    java.lang.String ID);
```

Parameters

ID

The ID used to represent the current time zone.

Example

The following example shows how to change the ID of the default time zone for the system.

```
// simpletimezone_setid.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create our own time zone named XYZ with an offset of
        // 100 milliseconds from GMT.  Set the daylight savings
        // time period from the first Sunday in April to the last
        // Sunday in October.  Make this the default time zone.
        SimpleTimeZone xyz = new SimpleTimeZone(100, "XYZ",
            Calendar.APRIL, 1, Calendar.SUNDAY, 2,
            Calendar.OCTOBER, -1, Calendar.SUNDAY, 2);
        TimeZone.setDefault(xyz);

        // Change the ID of this time zone to "WXY".
        xyz.setID("WXY");

        System.out.println("default time zone = " +
            TimeZone.getDefault().getID());
    }
}

/*
Output:
default time zone = WXY
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.setRawOffset Method

Sets the offset of the time zone from UTC, ignoring daylight savings time.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract void setRawOffset(
    int offsetMillis);
```

Parameters

offsetMillis

The offset, represented in milliseconds, of the time zone from UTC, ignoring daylight savings time. Areas east of UTC have positive offsets, while areas west of UTC have negative offsets.

Example

The following example changes the raw offset for a made-up time zone to 200 milliseconds off GMT.

```
// simpletimezone_setrawoffset.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create our own time zone named XYZ with an offset of
        // 100 milliseconds from GMT.  Set the daylight savings
        // time period from the first Sunday in April to the last
        // Sunday in October.
        SimpleTimeZone xyz = new SimpleTimeZone(100, "XYZ",
            Calendar.APRIL, 1, Calendar.SUNDAY, 2,
            Calendar.OCTOBER, -1, Calendar.SUNDAY, 2);
        TimeZone.setDefault(xyz);

        // Change the offset to 200 milliseconds.
        xyz.setRawOffset(200);

        System.out.println("raw offset: " + xyz.getRawOffset());
    }
}

/*
Output:
raw offset: 200
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TimeZone.useDaylightTime Method

Determines whether the current time zone uses daylight savings time.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public abstract boolean useDaylightTime();
```

Return Value

true if the current time zone uses daylight savings time; false otherwise.

Example

The following example determines whether a given time zone uses daylight savings time.

```
// simpletimezone_usedaylighttime.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        TimeZone tz = TimeZone.getDefault();
        String name = tz.getID();
        System.out.println(name + " uses DST? " +
            tz.useDaylightTime());
    }
}

/*
Output:
PST uses DST? true
*/
```

See Also

Reference

TimeZone Class

Concepts

TimeZone Members

java.util Package

TooManyListenersException Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.TooManyListenersException
    extends java.lang.Exception
```

Inheritance Hierarchy

java.lang.Object

Exception

java.lang.Throwable

java.lang.Exception

java.util.TooManyListenersException

See Also

Concepts

TooManyListenersException Members

java.util Package

TooManyListenersException Members

The following tables list the members exposed by the TooManyListenersException type.

Public Constructors

Name	Description
TooManyListenersException	Overloaded.

Public Properties (see also Protected Properties)

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from **Exception**)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

TooManyListenersException Class

Concepts

java.util Package

TooManyListenersException Constructor

Overload List

Name	Description
TooManyListenersException ()	
TooManyListenersException (String)	
TooManyListenersException (SerializationInfo, StreamingContext)	
TooManyListenersException (String, Exception)	

See Also

Reference

TooManyListenersException Class

Concepts

TooManyListenersException Members

java.util Package

TooManyListenersException Constructor ()

Initializes a new instance of the TooManyListenersException Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.TooManyListenersException();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

TooManyListenersException Class

Concepts

TooManyListenersException Members

java.util Package

TooManyListenersException Constructor (String)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.TooManyListenersException(
    java.lang.String s);
```

Parameters

s

See Also
Reference
TooManyListenersException Class
Concepts
TooManyListenersException Members
java.util Package

TooManyListenersException Constructor (SerializationInfo, StreamingContext)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.util.TooManyListenersException(
    System.Runtime.Serialization.SerializationInfo info,
    System.Runtime.Serialization.StreamingContext context);
```

Parameters

info

context

See Also

Reference

TooManyListenersException Class

Concepts

TooManyListenersException Members

java.util Package

TooManyListenersException Constructor (String, Exception)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.TooManyListenersException(
    java.lang.String s,
    System.Exception inner);
```

Parameters

s

inner

See Also
Reference
TooManyListenersException Class
Concepts
TooManyListenersException Members
java.util Package

TooManyListenersException Methods

Public Methods

Name	Description
equals	(inherited from Object)
fillInStackTrace	(inherited from Throwable)
GetBaseException	(inherited from Exception)
hashCode	(inherited from Object)
getLocalizedMessage	(inherited from Throwable)
getMessage	(inherited from Throwable)
GetObjectData	(inherited from Throwable)
getClass	(inherited from Object)
clone	(inherited from Throwable)
printStackTrace	Overloaded. (inherited from Throwable)
toString	(inherited from Throwable)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

TooManyListenersException Class

Concepts

java.util Package

TooManyListenersException Properties

Public Properties

Name	Description
HelpLink	(inherited from Exception)
InnerException	(inherited from **Exception**)
Message	(inherited from **Exception**)
Source	(inherited from **Exception**)
StackTrace	(inherited from **Exception**)
TargetSite	(inherited from **Exception**)

Protected Properties

Name	Description
HResult	(inherited from **Exception**)

See Also

Reference

TooManyListenersException Class

Concepts

java.util Package

TreeMap Class

Provides a sorted map object that implements the SortedMap interface.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.TreeMap
    extends java.util.AbstractMap
    implements java.util.SortedMap, java.lang.Cloneable, java.io.Serializable
```

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractMap

   java.util.TreeMap

See Also

Concepts

TreeMap Members

java.util Package

TreeMap Members

Provides a sorted map object that implements the SortedMap interface.

The following tables list the members exposed by the TreeMap type.

Public Constructors

Name	Description
TreeMap	Overloaded.

Public Methods

Name	Description
clear	Overridden. Removes all the mappings from a TreeMap object.
comparator	Retrieves the comparator that is used in ordering the values in a TreeMap object.
containsKey	Overridden. Checks if a TreeMap object contains a specific key.
containsValue	Overridden. Checks if a TreeMap object contains one or more keys that correspond to a specific value.
entrySet	Overridden. Retrieves the set of mappings contained in a TreeMap object.
equals	(inherited from AbstractMap)
firstKey	Retrieves the lowest key in a TreeMap object.
get	Overridden. Retrieves the value associated with a specific key in a TreeMap object.
hashCode	(inherited from AbstractMap)
getClass	(inherited from Object)
headMap	Retrieves a part of the map whose keys are less than a specific key value.
isEmpty	(inherited from AbstractMap)
keySet	Overridden. Retrieves the set of keys in a TreeMap object.
lastKey	Retrieves the highest key from a TreeMap object.
clone	
put	Overridden. Creates an association between an object and specified key entry in a TreeMap object.
putAll	Overridden. Copies the entries of a specific map to a TreeMap object.
remove	Overridden. Removes an entry from a TreeMap object.
size	Overridden. Retrieves the number of key-value mappings of a TreeMap object.
subMap	Creates a sub map that contains the objects within a specified range from a TreeMap object.

tailMap	Retrieves a partial map whose keys are greater than or equal to a specific key in a TreeMap object.
toString	(inherited from AbstractMap)
traverse	Traverses a TreeMap object. Obsolete.
values	Overridden. Retrieves a collection of values in a TreeMap object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

TreeMap Class

Concepts

java.util Package

TreeMap Constructor

Overload List

Name	Description
TreeMap()	Constructs a TreeMap object sorted according to the natural order.
TreeMap (Comparator)	Constructs a TreeMap object sorted by a Comparator object.
TreeMap (Map)	Constructs a TreeMap object that has the same mappings as the passed Map object.
TreeMap (SortedMap)	Constructs a TreeMap object that has the same mappings as the passed SortedMap object.

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap Methods

Public Methods

Name	Description
clear	Overridden. Removes all the mappings from a TreeMap object.
comparator	Retrieves the comparator that is used in ordering the values in a TreeMap object.
containsKey	Overridden. Checks if a TreeMap object contains a specific key.
containsValue	Overridden. Checks if a TreeMap object contains one or more keys that correspond to a specific value.
entrySet	Overridden. Retrieves the set of mappings contained in a TreeMap object.
equals	(inherited from AbstractMap)
firstKey	Retrieves the lowest key in a TreeMap object.
get	Overridden. Retrieves the value associated with a specific key in a TreeMap object.
hashCode	(inherited from AbstractMap)
getClass	(inherited from Object)
headMap	Retrieves a part of the map whose keys are less than a specific key value.
isEmpty	(inherited from AbstractMap)
keySet	Overridden. Retrieves the set of keys in a TreeMap object.
lastKey	Retrieves the highest key from a TreeMap object.
clone	
put	Overridden. Creates an association between an object and specified key entry in a TreeMap object.
putAll	Overridden. Copies the entries of a specific map to a TreeMap object.
remove	Overridden. Removes an entry from a TreeMap object.
size	Overridden. Retrieves the number of key-value mappings of a TreeMap object.
subMap	Creates a sub map that contains the objects within a specified range from a TreeMap object.
tailMap	Retrieves a partial map whose keys are greater than or equal to a specific key in a TreeMap object.
toString	(inherited from AbstractMap)
traverse	Traverses a TreeMap object. Obsolete.
values	Overridden. Retrieves a collection of values in a TreeMap object.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

TreeMap Class

Concepts

java.util Package

TreeMap.clear Method

Removes all the mappings from a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

Example

In this example, you create and initialize a TreeMap object, and then you clear it and display its elements. The result is an empty set.

```
// TreeMap-ctor3.jsl
// TreeMap.ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm1 = new TreeMap();

        // Add some elements:
        tm1.put("4","Sally Abolrous");
        tm1.put("1","Craig Combel");
        tm1.put("2","Pille Mandla");

        // Clear the map:
        tm1.clear();

        // Display the values of the cleared map:
        System.out.println("The values of tm1 are: " + tm1.values());
    }
}

/*
The values of tm1 are: []
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.clone Method

Creates a shallow copy of a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

The TreeMap copy.

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.comparator Method

Retrieves the comparator that is used in ordering the values in a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Comparator comparator();
```

Return Value

A comparator object used in ordering TreeMap.

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.containsKey Method

Checks if a TreeMap object contains a specific key.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean containsKey(
    java.lang.Object k);
```

Parameters

k

The key searched for in the TreeMap object.

Return Value

true if the key is found; false otherwise.

Example

In this example, you create and initialize a TreeMap object, and then you check if it contains the key "5." The result is true.

```
// TreeMap-contkey1.jsl
// TreeMap.containsKey example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("3","Craig Combel");
        tm.put("5","Pille Mandla");

        // Check if the key "5" is in the map:
        System.out.println(tm.containsKey("5"));
    }
}

/*
Output:
true
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.containsValue Method

Checks if a TreeMap object contains one or more keys that correspond to a specific value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean containsValue(
    java.lang.Object v);
```

Parameters

v

The value searched for in the TreeMap object.

Return Value

true if the value is found; false otherwise.

Example

In this example, you create and initialize a TreeMap object, and then you check if it contains the element " Pille Mandla." The result is true.

```
// TreeMap-contval.jsl
// TreeMap.containsValue example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("3","Craig Combel");
        tm.put("5","Pille Mandla");

        // Check if "Pille Mandla" is in the map:
        System.out.println(tm.containsValue("Pille Mandla"));
    }
}

/*
Output:
true
*/
```

See Also
Reference
TreeMap Class
Concepts
TreeMap Members
java.util Package

TreeMap.entrySet Method

Retrieves the set of mappings contained in a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Set entrySet();
```

Return Value

A set of mappings in the TreeMap object.

Example

In this example, you create and initialize a TreeMap object, and then you display the mappings contained in the object.

```
// TreeMap-entrysm1.jsl
// TreeMap.entrySet example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("3","Craig Combel");
        tm.put("5","Pille Mandla");

        // Display the entry set of the map:
        System.out.println("The entry set is:\n" + tm.entrySet());
    }
}

/*
Output:
The entry set is:
[[1, Sally Abolrous], [3, Craig Combel], [5, Pille Mandla]]
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.firstKey Method

Retrieves the lowest key in a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object firstKey();
```

Return Value

The lowest key in the TreeMap object.

Example

In this example, you create and initialize a TreeMap object, and then you display the lowest key in the object.

```
// TreeMap-firstK1.jsl
// TreeMap.firstKey example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("3","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("5","Pille Mandla");

        // Display the lowest key:
        System.out.println("The lowest key value is: " +
                           tm.firstKey());
    }
}

/*
Output:
The lowest key value is: 2
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.get Method

Retrieves the value associated with a specific key in a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object get(
    java.lang.Object k);
```

Parameters

k

The key value associated with the object searched for in the TreeMap object.

Return Value

The object at the specified key entry in TreeMap.

Example

In this example, you create and initialize a TreeMap object, and then you display the element associated with the key "2."

```
// TreeMap-get1.jsl
// TreeMap.get example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1", "Sally Abolrous");
        tm.put("2","Craig Combel");

        // Display the element that corresponds to the key #2:
        System.out.println("The element #2 is: " + tm.get("2"));
    }
}

/*
Output:
The element #2 is: Craig Combel
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.headMap Method

Retrieves a part of the map whose keys are less than a specific key value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.SortedMap headMap(
    java.lang.Object toV);
```

Parameters

toV

The highest key value.

Return Value

A part of the map whose keys are less than toV.

Example

In this example, you create and initialize a TreeMap object, and then you display the elements of headMap.

```
// TreeMap-headm1.jsl
// TreeMap.headMap example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("3","Craig Combel");
        tm.put("5","Pille Mandla");

        // Display the headMap:
        System.out.println("The head map is:\n" + tm.headMap("5"));
    }
}

/*
Output:
The head map is:
{1=Sally Abolrous, 3=Craig Combel}
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.keySet Method

Retrieves the set of keys in a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Set keySet();
```

Return Value

The set of keys contained in the TreeMap object.

Example

In this example, you create and initialize a TreeMap object, and then you display the keySet contained in the object.

```
// TreeMap-keyset1.jsl
// TreeMap.keySet example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("3","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("5","Pille Mandla");

        // Remove the key set:
        System.out.println("The key set is: " + tm.keySet());
    }
}

/*
Output:
The key set is: [2, 3, 5]
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.lastKey Method

Retrieves the highest key from a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object lastKey();
```

Return Value

The highest key object retrieved from the TreeMap object.

Example

In this example, you create and initialize a TreeMap object, and then you display the highest key in the object.

```
// TreeMap-lastK1.jsl
// TreeMap.lastKey example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("3","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("5","Pille Mandla");

        // Display the highest key:
        System.out.println("The highest key value is: " +
                           tm.lastKey());
    }
}

/*
Output:
The highest key value is: 5
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.put Method

Creates an association between an object and specified key entry in a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object put(
    java.lang.Object k,
    java.lang.Object v);
```

Parameters

k

The key with which the object is associated in the TreeMap object.

v

The object associated with the specified key in the TreeMap object.

Return Value

The object associated with the specified key in TreeMap.

Example

In this example, you create and initialize a TreeMap object, and then you display element at the key number "2."

```
// TreeMap-put1.jsl
// TreeMap.put example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tMap = new TreeMap();

        // Add some elements:
        tMap.put("1", "Sam Abolrous");
        tMap.put("2","Camelia Solomon");

        // Display the element that corresponds to the key #2:
        System.out.println("The element #2 is: " + tMap.get("2"));
    }
}

/*
Output:
The element #2 is: Camelia Solomon
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.putAll Method

Copies the entries of a specific map to a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void putAll(
    java.util.Map m);
```

Parameters

m

The map to be copied.

Example

In this example, you create and initialize a TreeMap object, and then you copy it to another TreeMap object.

```
// TreeMap-putAll1.jsl
// TreeMap.putAll example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create two TreeMap objects:
        TreeMap tm1 = new TreeMap();
        TreeMap tm2 = new TreeMap();

        // Add some elements to tm1:
        tm1.put("1","Sally Abolrous");
        tm1.put("2","Craig Combel");

        // Copy tm1 to tm2:
        tm2.putAll(tm1);

        // Display the element #2 in both objects:
        System.out.println("The element #2 in tm1 is: "
                           + tm1.get("2"));
        System.out.println("The element #2 in tm2 is: "
                           + tm2.get("2"));
    }
}

/*
Output:
The element #2 in tm1 is: Craig Combel
The element #2 in tm2 is: Craig Combel
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.remove Method

Removes an entry from a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object remove(
    java.lang.Object k);
```

Parameters

k

The key at which the entry is removed.

Return Value

The removed entry object in TreeMap.

Example

In this example, you create and initialize a TreeMap object, and then you remove one element from the object.

```
// TreeMap-rem1.jsl
// TreeMap.remove example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("3","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("5","Pille Mandla");

        // Remove an object:
        System.out.println("The following object has been removed: " +
                           tm.remove("5"));
    }
}

/*
Output:
The following object has been removed: Pille Mandla
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.size Method

Retrieves the number of key-value mappings of a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

Return Value

An integer that represents number of key-value mappings in the TreeMap object.

Example

In this example, you create and initialize a TreeMap object, and then you display the number of the key-value mappings of the object.

```
// TreeMap-size1.jsl
// TreeMap.size example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("3","Pille Mandla");

        // Display the size of the TreeMap:
        System.out.println("The size is: " + tm.size());
    }
}

/*
Output:
The size is: 3
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.subMap Method

Creates a sub map that contains the objects within a specified range from a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.SortedMap subMap(
    java.lang.Object fromV,
    java.lang.Object toV);
```

Parameters

fromV

The starting key in the TreeMap.

toV

The ending key in the TreeMap.

Return Value

A sorted map that contains the range of objects between the starting and ending keys in the TreeMap object.

Example

In this example, you create and initialize a TreeMap object, and then you create a subMap object from the original object.

```
// TreeMap-submap1.jsl
// TreeMap.subMap example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("3","Pille Mandla");

        // Display the SortedMap:
        System.out.println(tm.subMap("1","3"));
    }
}

/*
Output:
{1=Sally Abolrous, 2=Craig Combel}
*/
```

Remarks

The starting key is included in the range, but the ending key is excluded.

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

TreeMap.tailMap Method

Retrieves a partial map whose keys are greater than or equal to a specific key in a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.SortedMap tailMap(
    java.lang.Object fromV);
```

Parameters

fromV

The lowest key of the tailMap object

Return Value

A sorted map that contains the entries greater than or equal to fromV.

Example

In this example, you create and initialize a TreeMap object, and then you create a tailMap object from the original object.

```
// TreeMap-tailm1.jsl
// TreeMap.tailMap example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("3","Pille Mandla");

        // Display the tailMap:
        System.out.println("The tail map is:\n" + tm.tailMap("2"));
    }
}

/*
Output:
The tail map is:
{2=Craig Combel, 3=Pille Mandla}
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.traverse Method

NOTE: This Method is now obsolete.

Traverses a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void traverse();
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeMap.values Method

Retrieves a collection of values in a TreeMap object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Collection values();
```

Return Value

The collection of values contained in the TreeMap object.

Example

In this example, you create and initialize a TreeMap object, and then you display the values contained in the object.

```
// TreeMap-vla1.jsl
// TreeMap.values example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a TreeMap object:
        TreeMap tm = new TreeMap();

        // Add some elements:
        tm.put("1","Sally Abolrous");
        tm.put("2","Craig Combel");
        tm.put("3","Pille Mandla");

        // Display the elements in the TreeMap object:
        System.out.println("The values are:\n" + tm.values());
    }
}

/*
Output:
The values are:
[Sally Abolrous, Craig Combel, Pille Mandla]
*/
```

See Also

Reference

TreeMap Class

Concepts

TreeMap Members

java.util Package

TreeSet Class

A class that represents a sorted set of elements. The elements are sorted using the comparator provided through constructor or using the Comparable interface methods implemented by the elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.TreeSet
    extends java.util.AbstractSet
    implements java.util.SortedSet, java.lang.Cloneable, java.io.Serializable
```

Remarks

It is a requirement that all elements of the set must implement Comparable.

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractCollection

    java.util.AbstractSet

     java.util.TreeSet

See Also

Concepts

TreeSet Members

java.util Package

TreeSet Members

A class that represents a sorted set of elements. The elements are sorted using the comparator provided through constructor or using the Comparable interface methods implemented by the elements.

The following tables list the members exposed by the TreeSet type.

Public Constructors

Name	Description
TreeSet	Overloaded. Creates a new TreeSet object.

Public Methods

Name	Description
add	Overridden. Inserts an element into a TreeSet object.
addAll	(inherited from AbstractCollection)
clear	Overridden. Removes all the elements from a TreeSet object.
comparator	Returns the comparator used in sorting a TreeSet object.
contains	Overridden. Checks if a TreeSet object contains a specified element.
containsAll	(inherited from AbstractCollection)
equals	(inherited from AbstractSet)
first	Returns the first element from the sorted list of elements in a TreeSet object.
hashCode	(inherited from AbstractSet)
getClass	(inherited from Object)
headSet	Returns a subset of elements in a TreeSet instance that precede a specified value.
isEmpty	Overridden. Checks whether a TreeSet object is empty.
iterator	Overridden. Returns an iterator over a TreeSet object.
last	Returns the last element from the sorted list of elements in a TreeSet object.
clone	
remove	Overridden. Removes a specified element from a TreeSet object.
removeAll	(inherited from AbstractCollection)
retainAll	(inherited from AbstractCollection)
size	Overridden. Retrieves the number of elements in a TreeSet object.
subSet	Retrieves a partial sorted set from a TreeSet object whose elements are within a specific range.

tailSet	Returns a subset of elements in a TreeSet object that follow a specified value.
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)
traverse	Not supported. Obsolete.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

TreeSet Class

Concepts

java.util Package

TreeSet Constructor

Creates a new TreeSet object.

Overload List

Name	Description
TreeSet ()	Creates a new TreeSet object that contains its elements sorted according to the natural order.
TreeSet (Collection)	Creates a new TreeSet object that contains elements of a specified collection sorted according to the natural order.
TreeSet (Comparator)	Creates a new TreeSet object whose elements are sorted according to a specific comparator.
TreeSet (SortedSet)	Creates a new TreeSet object that contains the elements of a specified set sorted according to the same order.

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet Constructor ()

Creates a new TreeSet object that contains its elements sorted according to the natural order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.TreeSet();
```

Example

```
// Create a new TreeSet object:
TreeSet ts = new TreeSet();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet Constructor (Collection)

Creates a new TreeSet object that contains elements of a specified collection sorted according to the natural order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.TreeSet(
    java.util.Collection c);
```

Parameters

c

The collection of elements that represents the TreeSet elements.

Example

```
// TS-ctor2.jsl
// TreeSet.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
         // Create and initialize a SotedSet collection:
         Set  ss = new TreeSet();
         ss.add(new Integer(3));
         ss.add(new Integer(7));
         ss.add(new Integer(5));

        // Create a TreeSet object using the SotedSet collection:
         TreeSet ts = new TreeSet(ss);
         System.out.println(ts);
    }
}

/*
Output:
[3, 5, 7]
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet Constructor (Comparator)

Creates a new TreeSet object whose elements are sorted according to a specific comparator.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.TreeSet(
    java.util.Comparator comp);
```

Parameters

comp

The comparator used to sort the elements in the TreeSet object.

Example

See the example on comparator.

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet Constructor (SortedSet)

Creates a new TreeSet object that contains the elements of a specified set sorted according to the same order.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.TreeSet(
    java.util.SortedSet s);
```

Parameters

s

The set that contains the elements of the TreeSet object.

Example

```
// TS-ctor4.jsl
// TreeSet.#ctor example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
         // Create and initialize a Set collection:
         SortedSet  ss = new TreeSet();
         ss.add(new Integer(2));
         ss.add(new Integer(6));
         ss.add(new Integer(4));

        // Create a TreeSet object using the SotedSet collection:
         TreeSet ts = new TreeSet(ss);
         System.out.println(ts);
    }
}

/*
Output:
[2, 4, 6]
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet Methods

Public Methods

Name	Description
add	Overridden. Inserts an element into a TreeSet object.
addAll	(inherited from AbstractCollection)
clear	Overridden. Removes all the elements from a TreeSet object.
comparator	Returns the comparator used in sorting a TreeSet object.
contains	Overridden. Checks if a TreeSet object contains a specified element.
containsAll	(inherited from AbstractCollection)
equals	(inherited from AbstractSet)
first	Returns the first element from the sorted list of elements in a TreeSet object.
hashCode	(inherited from AbstractSet)
getClass	(inherited from Object)
headSet	Returns a subset of elements in a TreeSet instance that precede a specified value.
isEmpty	Overridden. Checks whether a TreeSet object is empty.
iterator	Overridden. Returns an iterator over a TreeSet object.
last	Returns the last element from the sorted list of elements in a TreeSet object.
clone	
remove	Overridden. Removes a specified element from a TreeSet object.
removeAll	(inherited from AbstractCollection)
retainAll	(inherited from AbstractCollection)
size	Overridden. Retrieves the number of elements in a TreeSet object.
subSet	Retrieves a partial sorted set from a TreeSet object whose elements are within a specific range.
tailSet	Returns a subset of elements in a TreeSet object that follow a specified value.
toArray	Overloaded. (inherited from AbstractCollection)
toString	(inherited from AbstractCollection)
traverse	Not supported. Obsolete.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

TreeSet Class

Concepts

java.util Package

TreeSet.add Method

Inserts an element into a TreeSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean add(
    java.lang.Object e);
```

Parameters

e

The element to be inserted into the TreeSet object.

Return Value

true if the element is added successfully; false otherwise.

Example

```
// TS-add1.jsl
// TreeSet.add example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Create new element objects:
        Integer x = new Integer(5);
        Integer y = new Integer(4);

        // Add the elements to the set and display it:
        System.out.println(ts.add(x));
        System.out.println(ts.add(y));
    }
}
/*
Output:
true
true
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.clear Method

Removes all the elements from a TreeSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

Example

In this example, you create a TreeSet object, and then you clear it and display the set before and after the clearing.

```
// TS-clear.jsl
// TreeSet.clear example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Create a new TreeSet object:
        TreeSet ts =  new TreeSet();

        // Create new element objects:
        Integer x = new Integer(52);
        Integer y = new Integer(43);
        Integer z = new Integer(45);


        // Add some elements:
        ts.add(x);
        ts.add(y);
        ts.add(z);

        // Display ts:
        System.out.println(ts);

        // Clear the set:
        ts.clear();

        // Display ts after clearing:
        System.out.println(ts);
    }
}
/*
Output:
[43, 45, 52]
[]
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.comparator Method

Returns the comparator used in sorting a TreeSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Comparator comparator();
```

Return Value

A comparator or null if the TreeSet elements are sorted according to the natural order.

Example

```
// TS-Comp1.jsl
// TreeSet.COmparator example

import java.util.*;

public class x
{
    public static void main()
    {
        TreeSet empAgeSet = new TreeSet(new EmployeeComparator_Age());
        String[] empNames =
                {"Dilbert","Murphy","Don","Wattson","Noddy"};
        Random rnd = new Random();

        for(int i=0; i<empNames.length; i++)
        {
            empAgeSet.add(new Employee(empNames[i], 18+rnd.nextInt(42),
                        rnd.nextInt(Integer.MAX_VALUE)));
        }
        Iterator empIterator = empAgeSet.iterator();

        System.out.println("Name\tAge\tPIN");

        while(empIterator.hasNext())
        {
            System.out.println(empIterator.next());
        }
    }
}

class EmployeeComparator_Age implements Comparator
{
    public int compare(Object emp1, Object emp2)
    {
        Employee e1 = (Employee)emp1;
        Employee e2 = (Employee)emp2;
        int ret = 1;

        if(e1.age == e2.age) ret = 0;
        if(e1.age < e2.age) ret = -1;

        return ret;
    }
}

class Employee
{
    String name;
    int    age;
```

```
    int    pinNumber;
    public Employee(String n, int a, int s)
    {
        name = n;
        age = a;
        pinNumber = s;
    }

    public String toString()
    {
        return name + "\t" + age + "\t" + pinNumber;
    }
}

/*
Output:
Wattson, 19, 1570241606
Dilbert, 24, 1055022363
Noddy, 31, 1946154285
Einstein, 52, 467910070
Murphy, 56, 782476970
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.contains Method

Checks if a TreeSet object contains a specified element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean contains(
    java.lang.Object e);
```

Parameters

e

The element to search for.

Return Value

true if the element is found; false otherwise.

Example

In this example, you create and initialize a TreeSet object, and then you check to see if it contains a specified element "x".

```
// TS-cont1.jsl
// TreeSet.contains example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Create a new TreeSet object:
        TreeSet ts =  new TreeSet();

        // Create new element objects:
        Integer x = new Integer(52);
        Integer y = new Integer(43);
        Integer z = new Integer(45);

        // Add the elements to the set:
        ts.add(x);
        ts.add(y);
        ts.add(z);

        // Check if the set contains "x":
        System.out.println(ts.contains(x));
    }
}
/*
Output:
true
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.first Method

Returns the first element from the sorted list of elements in a TreeSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object first();
```

Return Value

The first element in the TreeSet object.

Example

```
// TS-first.jsl
// TreeSet.first example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Add some elements:
        ts.add(new Integer(5));
        ts.add(new Integer(4));
        ts.add(new Integer(9));

        // Display the first and the last:
        System.out.println("The first element: " + ts.first());
        System.out.println("The last element: " + ts.last());
    }
}
/*
Output:
The first element: 4
The last element: 9
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.headSet Method

Returns a subset of elements in a TreeSet instance that precede a specified value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.SortedSet headSet(
    java.lang.Object toV);
```

Parameters

toV

The element to which the elements of the returned set are compared.

Return Value

A sorted set whose elements precede toV.

Example

In this example, you create and initialize a TreeSet object, and then you display the headSet and the tailSet objects.

```
// TS-hdset1.jsl
// TreeSet.headSet example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Add some elements:
        ts.add(new Integer(5));
        ts.add(new Integer(4));
        ts.add(new Integer(9));

        // Display the headSet and the tailSet:
        System.out.println("The headSet: " + ts.headSet(ts.last()));
        System.out.println("The tailSet: " + ts.tailSet(ts.last()));
    }
}

/*
Output:
The headSet: [4,5]
The tailSet: [9]
*/
```

Remarks

The returned set does not include the value toV.

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.isEmpty Method

Checks whether a TreeSet object is empty.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isEmpty();
```

Return Value

true if the TreeSet object is empty; false otherwise.

Example

In this example, you create a TreeSet object, and then you clear it and display the set before and after the clearing.

```
// TS-isEmp1.jsl
// TreeSet.isEmpty example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Create a new TreeSet object:
        TreeSet ts =  new TreeSet();

        // Create new element objects:
        Integer x = new Integer(52);
        Integer y = new Integer(43);
        Integer z = new Integer(45);


        // Add some elements:
        ts.add(x);
        ts.add(y);
        ts.add(z);

        // Display ts:
        System.out.println("The original set: "  + ts);

        // Clear the set:
        ts.clear();

        // Check to see if the set is empty and
        // Display the empty set:
        if (ts.isEmpty())
            System.out.println("The empty set: " + ts);
    }
}
/*
Output:
The original set: [43, 45, 52]
The empty set: []
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

TreeSet.iterator Method

Returns an iterator over a TreeSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Iterator iterator();
```

Return Value

An iterator to be used over the TreeSet object.

Example

In this example, you create and initialize a TreeSet object, and then you retrieve an iterator to iterate over the elements and display them.

```
// TS-iter1.jsl
// TreeSet.iterator example

import java.util.*;

public class MyClass
{
    public static void main()
    {
        // Create a new TreeSet object:
        TreeSet ts =  new TreeSet();

        // Create new element objects:
        Integer x = new Integer(9);
        Integer y = new Integer(8);
        Integer z = new Integer(7);

        // Add some elements:
        ts.add(x);
        ts.add(y);
        ts.add(z);

        // Retrieve an iterator to the set:
        Iterator iter = ts.iterator();

        // Extract and display the elements from the set.
        // Note that the elements may not follow the order in which
        //  they weree added to the Set.

        while(iter.hasNext())
        {
            System.out.print(iter.next() + " ");
            iter.remove();
        }
    }
}
/*
Output:
7 8 9
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members
java.util Package

TreeSet.last Method

Returns the last element from the sorted list of elements in a TreeSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object last();
```

Return Value

The last element from the sorted list of elements in the TreeSet object.

Example

See the example under first.

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.remove Method

Removes a specified element from a TreeSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean remove(
    java.lang.Object e);
```

Parameters

e

The element to be removed from the TreeSet object.

Return Value

true if the element is successfully removed; false otherwise.

Example

See the example under size.

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.size Method

Retrieves the number of elements in a TreeSet object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

Return Value

The number of elements in the TreeSet object.

Example

In this example, you create and initialize a TreeSet object, and then you remove one element and display the size before and after the removal.

```
// TS-size1.jsl
// TreeSet.size example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Declare some Integer objects:
        Integer x = new Integer(5);
        Integer y = new Integer(213);
        Integer z = new Integer(4);

        // Add some elements:
        ts.add(x);
        ts.add(y);
        ts.add(z);

        // Display the size of the TreeSet object:
        System.out.println("The size is: " + ts.size());

        // Remove one element:
        ts.remove(x);

        // Display the size of the TreeSet object:
        System.out.println("The new size is: " + ts.size());
    }
}

/*
Output:
The size is: 3
The new size is: 2
*/
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.subSet Method

Retrieves a partial sorted set from a TreeSet object whose elements are within a specific range.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.SortedSet subSet(
    java.lang.Object fromV,
    java.lang.Object toV);
```

Parameters

fromV

The lowest value in the range.

toV

The highest value in the range.

Return Value

A sorted set that contains the range from fromV to toV.

Example

In this example, you create and initialize a TreeSet object, and then you retrieve and display two subsets.

```
// TS-subs1.jsl
// TreeSet.subset example

import java.util.*;

public class MyClass
{
    public static void main(String[] args)
    {
        // Create a new TreeSet object:
        TreeSet ts = new TreeSet();

        // Declare some Double objects:
        Double x = new Double(5.1);
        Double y = new Double(7.3);
        Double z = new Double(9.2);

        // Add some elements:
        ts.add(x);
        ts.add(y);
        ts.add(z);

        System.out.println("The TreeSet elements are: " +
                           "x=" + x + ", y=" + y + ", z=" + z);
        // Display the subsets between two different elements:
        System.out.println("The subset from x to z is: " +
                           ts.subSet(x,z));
        System.out.println("The subset from x to y is: " +
                           ts.subSet(x,y));
    }
}

/*
Output:
The TreeSet elements are: x=5.1, y=7.3, z=9.2
The subset from x to z is: [5.1,7.3]
The subset from x to y is: [5.1]
```

```
*/
```

Remarks

The fromV value is included in the range, while the toV value is excluded.

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.tailSet Method

Returns a subset of elements in a TreeSet object that follow a specified value.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.SortedSet tailSet(
    java.lang.Object fromV);
```

Parameters

fromV

The element to which the elements of the returned set are compared.

Return Value

A subset whose elements follow fromV.

Example

See the example under headSet.

Remarks

The returned subset includes the value fromV.

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

TreeSet.traverse Method

NOTE: This Method is now obsolete.

Not supported.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void traverse();
```

See Also

Reference

TreeSet Class

Concepts

TreeSet Members

java.util Package

Vector Class (J#)

Represents a dynamic array of elements. Unlike a static array, a vector will grow in size once it has reached capacity.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.Vector
    extends java.util.AbstractList
    implements java.lang.Cloneable, java.util.List, java.io.Serializable
```

Example

The following example demonstrates the add, clear, contains, elementAt, elements, indexOf, isEmpty, remove, set, and size methods of the Vector class.

```
// vector_overview.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("One");
        v.add("Two");
        v.add("Three");

        // Is there an element "Two" in the Vector?
        boolean isTwo = v.contains("Two");
        System.out.println("v contains \"Two\"? " + isTwo);

        // What element is at index 2?
        String s = (String)v.elementAt(2);
        System.out.println("v[2] = " + s);

        // What index is element "One" at?
        int idx = v.indexOf("One");
        System.out.println("\"One\" is at index " + idx);

        // How many elements are in the Vector?
        if (!v.isEmpty())
        {
            System.out.println("size = " + v.size());
        }

        // Change the element at index 1 to "Four".
        Object old1 = v.set(1, "Four");
        System.out.println("Removed element " + old1.toString());

        // Enumerate over the elements in the Vector.
        Enumeration e = v.elements();
        while (e.hasMoreElements())
        {
            System.out.println(e.nextElement());
        }

        // Remove the last element in the Vector.
        s = (String)v.remove(v.size() - 1);
        System.out.println("Removed element " + s);
```

```
        // Clear all elements in the Vector.
        v.clear();
    }
}

/*
Output:
v contains "Two"? true
v[2] = Three
"One" is at index 0
size = 3
Removed element Two
One
Four
Three
Removed element Three
*/
```

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractCollection

    java.util.AbstractList

      java.util.Vector

        java.util.Stack

See Also

Concepts

Vector Members

java.util Package

Vector Members (J#)

Represents a dynamic array of elements. Unlike a static array, a vector will grow in size once it has reached capacity.

The following tables list the members exposed by the Vector type.

Public Constructors

Name	Description
Vector	Overloaded. Initializes a new instance of a Vector object.

Public Fields

Name	Description
capacityIncrement	A value representing the size to increase the vector once its capacity is reached.
elementCount	The number of elements currently stored in the vector.
elementData	An internal static array containing the elements of the vector.
modCount	(inherited from AbstractList)

Public Methods

Name	Description
add	Overloaded. Overridden. Adds an element to the vector.
addAll	Overloaded. Overridden. Adds multiple elements to the vector at one time.
addElement	Adds an element to the end of the vector. The vector is automatically grown if there is insufficient room to add the element.
capacity	Returns the number of elements capable of being stored in the vector.
clear	Overridden. Removes all elements from the vector.
contains	Overridden. Determines whether the vector contains a given element.
containsAll	Overridden. Determines whether all the elements in the given collection are contained in the vector.
copyInto	Copies all the elements in the vector into an array.
elementAt	Returns the element at the specified index. The element is not removed from the vector.
elements	Returns an Enumeration object that can be used to efficiently traverse the elements in the vector.
ensureCapacity	Determines whether the vector can hold the given number of elements. If it cannot, a new vector is created that is large enough to store the desired number of elements and the contents of the old vector are copied into the newly created vector.
equals	Overridden. Determines whether two vectors are equal. Two vectors are considered equal if they contain e xactly the same elements.
firstElement	Returns the first element in the vector. The element is not removed from the vector.

get	Overridden. Returns the element at a given index. The element is not removed from the vector.
hashCode	Overridden. Provides a hash value representing this vector.
getClass	(inherited from Object)
indexOf	Overloaded. Overridden. Returns the index of the first occurrence of a given element.
insertElementAt	Inserts an element into the vector at the given index. The vector is automatically grown if there is insufficient room to add the element.
isEmpty	Overridden. Determines whether the vector contains any elements.
iterator	(inherited from AbstractList)
lastElement	Returns the last element in the vector. The element is not removed from the vector.
lastIndexOf	Overloaded. Overridden. Returns the index of the last occurrence of a given element.
listIterator	Overloaded. (inherited from AbstractList)
clone	Creates a new instance of a Vector object that is a shallow copy of the vector.
remove	Overloaded. Overridden. Removes an element from the vector.
removeAll	Overridden. Removes all the elements contained in a collection from the vector. If an element appears more than once in the vector, all instances are removed. The elements in the vector beyond these elements are then shifted down.
removeAllElements	Removes all elements from the vector.
removeElement	Removes the first occurrence of an element from the vector. The elements in the vector beyond this element are then shifted down.
removeElementAt	Removes the element from the vector at the specified index. The elements in the vector beyond this index are then shifted down.
removeRange	Overridden. Removes all the elements from the vector starting with the element at fromIx to the element at one less than toIx.
retainAll	Overridden. Removes all elements from the vector that are not contained in the given collection.
set	Overridden. Sets the element at the given index to the given element.
setElementAt	Sets the element at the given index to the given element.
setSize	Sets the size of the vector to the given value. If the given size is greater than the current size of the vector, then the contents of the old vector are copied into a new vector of the given size. If the given size is less than the current size of the vector, then some elements will be removed from the vector.
size	Overridden. Returns the number of elements in the vector.
subList	Overridden. Returns a List of all the elements in the vector starting with the element at fromIx to the element at one less than toIx.

toArray	Overloaded. Overridden. Converts a vector into a static array.
toString	Overridden. Displays a human-readable representation of a vector.
trimToSize	Changes the capacity of a vector to match its size.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Vector Class

Concepts

java.util Package

Vector Fields

Public Fields

Name	Description
capacityIncrement	A value representing the size to increase the vector once its capacity is reached.
elementCount	The number of elements currently stored in the vector.
elementData	An internal static array containing the elements of the vector.
modCount	(inherited from AbstractList)

See Also

Reference

Vector Class

Concepts

java.util Package

Vector.capacityIncrement Field

A value representing the size to increase the vector once its capacity is reached.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected int capacityIncrement;
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.elementCount Field

The number of elements currently stored in the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected int elementCount;
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.elementData Field

An internal static array containing the elements of the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected java.lang.Object[] elementData;
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector Constructor

Initializes a new instance of a Vector object.

Overload List

Name	Description
Vector ()	Initializes a new instance of a Vector object.
Vector (Collection)	Initializes a new instance of a Vector object with the elements in the given collection.
Vector (int)	Initializes a new instance of a Vector object to the given size.
Vector (int, int)	Initializes a new instance of a Vector object to the given size. You can also specify an amount representing the size to increment the vector once it reaches capacity.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector Constructor ()

Initializes a new instance of a Vector object.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Vector();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector Constructor (Collection)

Initializes a new instance of a Vector object with the elements in the given collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Vector(
    java.util.Collection c);
```

Parameters

c

A collection whose elements are to be copied into the vector. The initial size of the vector is set to the same size as this collection. Only a shallow copy of these elements is performed.

See Also
Reference
Vector Class
Concepts
Vector Members
java.util Package

Vector Constructor (Int32)

Initializes a new instance of a Vector object to the given size.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Vector(
    int initSize);
```

Parameters

initSize

The initial size of the vector.

See Also
Reference
Vector Class
Concepts
Vector Members
java.util Package

Vector Constructor (Int32, Int32)

Initializes a new instance of a Vector object to the given size. You can also specify an amount representing the size to increment the vector once it reaches capacity.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Vector(
    int initSize,
    int increment);
```

Parameters

initSize

The initial size of the vector.

increment

The amount to increase the vector once it reaches capacity.

See Also
Reference
Vector Class
Concepts
Vector Members
java.util Package

Vector Methods

Public Methods

Name	Description
add	Overloaded. Overridden. Adds an element to the vector.
addAll	Overloaded. Overridden. Adds multiple elements to the vector at one time.
addElement	Adds an element to the end of the vector. The vector is automatically grown if there is insufficient room to add the element.
capacity	Returns the number of elements capable of being stored in the vector.
clear	Overridden. Removes all elements from the vector.
contains	Overridden. Determines whether the vector contains a given element.
containsAll	Overridden. Determines whether all the elements in the given collection are contained in the vector.
copyInto	Copies all the elements in the vector into an array.
elementAt	Returns the element at the specified index. The element is not removed from the vector.
elements	Returns an Enumeration object that can be used to efficiently traverse the elements in the vector.
ensureCapacity	Determines whether the vector can hold the given number of elements. If it cannot, a new vector is created that is large enough to store the desired number of elements and the contents of the old vector are copied into the newly created vector.
equals	Overridden. Determines whether two vectors are equal. Two vectors are considered equal if they contain e xactly the same elements.
firstElement	Returns the first element in the vector. The element is not removed from the vector.
get	Overridden. Returns the element at a given index. The element is not removed from the vector.
hashCode	Overridden. Provides a hash value representing this vector.
getClass	(inherited from Object)
indexOf	Overloaded. Overridden. Returns the index of the first occurrence of a given element.
insertElementAt	Inserts an element into the vector at the given index. The vector is automatically grown if there is insufficie nt room to add the element.
isEmpty	Overridden. Determines whether the vector contains any elements.
iterator	(inherited from AbstractList)
lastElement	Returns the last element in the vector. The element is not removed from the vector.
lastIndexOf	Overloaded. Overridden. Returns the index of the last occurrence of a given element.

listIterator	Overloaded. (inherited from AbstractList)
clone	Creates a new instance of a Vector object that is a shallow copy of the vector.
remove	Overloaded. Overridden. Removes an element from the vector.
removeAll	Overridden. Removes all the elements contained in a collection from the vector. If an element appears more than once in the vector, all instances are removed. The elements in the vector beyond these elements are then shifted down.
removeAllElements	Removes all elements from the vector.
removeElement	Removes the first occurrence of an element from the vector. The elements in the vector beyond this element are then shifted down.
removeElementAt	Removes the element from the vector at the specified index. The elements in the vector beyond this index are then shifted down.
removeRange	Overridden. Removes all the elements from the vector starting with the element at fromIx to the element at one less than toIx.
retainAll	Overridden. Removes all elements from the vector that are not contained in the given collection.
set	Overridden. Sets the element at the given index to the given element.
setElementAt	Sets the element at the given index to the given element.
setSize	Sets the size of the vector to the given value. If the given size is greater than the current size of the vector, then the contents of the old vector are copied into a new vector of the given size. If the given size is less than the current size of the vector, then some elements will be removed from the vector.
size	Overridden. Returns the number of elements in the vector.
subList	Overridden. Returns a List of all the elements in the vector starting with the element at fromIx to the element at one less than toIx.
toArray	Overloaded. Overridden. Converts a vector into a static array.
toString	Overridden. Displays a human-readable representation of a vector.
trimToSize	Changes the capacity of a vector to match its size.

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

Vector Class

Concepts

java.util Package

Vector.add Method

Adds an element to the vector.

Overload List

Name	Description
Vector.add (Object)	Adds an element to the end of the vector. The vector is automatically grown if there is insufficient room to add the element.
Vector.add (int, Object)	Adds an element to the vector at the specified index. The vector is automatically grown if there is insufficient room to add the element.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.add Method (Object)

Adds an element to the end of the vector. The vector is automatically grown if there is insufficient room to add the element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean add(
    java.lang.Object e);
```

Parameters

e

The element to add to the end of the vector.

Return Value

true if the element was successfully added to the vector; false otherwise.

Example

The following example creates a Vector object and appends three elements to it.

```
// vector_add.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("One");
        v.add("Two");
        v.add("Three");

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
One
Two
Three
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.add Method (Int32, Object)

Adds an element to the vector at the specified index. The vector is automatically grown if there is insufficient room to add the element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void add(
    int ix,
    java.lang.Object e);
```

Parameters

ix

The position within the vector to add the element.

e

The element to add to the vector.

Example

The following example creates a Vector object and adds three elements to it at the specified indices.

```
// vector_add_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        // Order should be "Orange", "Banana", "Apple".
        Vector v = new Vector();
        v.add(0, "Apple");
        v.add(0, "Orange");
        v.add(1, "Banana");

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
Orange
Banana
Apple
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.addAll Method

Adds multiple elements to the vector at one time.

Overload List

Name	Description
Vector.addAll (Collection)	Adds all the elements in the collection to the vector. The vector is automatically grown if there is insufficient room to add the elements.
Vector.addAll (int, Collection)	Adds all the elements in the collection to the vector starting at the specified index. The vector is automatically grown if there is insufficient room to add the elements.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.addAll Method (Collection)

Adds all the elements in the collection to the vector. The vector is automatically grown if there is insufficient room to add the elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean addAll(
    java.util.Collection c);
```

Parameters

c

The elements to be added to the collection. Only a shallow copy of these elements is performed.

Return Value

true if all the elements were successfully added to the vector; false otherwise.

Example

The following example adds all the elements of a LinkedList object into a Vector object.

```
// vector_addall.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a LinkedList and add three elements to it.
        LinkedList ll = new LinkedList();
        ll.add("Linked List Element 1");
        ll.add("Linked List Element 2");
        ll.add("Linked List Element 3");

        // Create a Vector containing the same elements as the
        // LinkedList.
        Vector v = new Vector(ll);

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
Linked List Element 1
Linked List Element 2
Linked List Element 3
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

Vector.addAll Method (Int32, Collection)

Adds all the elements in the collection to the vector starting at the specified index. The vector is automatically grown if there is insufficient room to add the elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean addAll(
    int ix,
    java.util.Collection c);
```

Parameters

ix

The position within the vector to add the first element. All subsequent elements in the collection are added immediately after this index.

c

The elements to be added to the collection. Only a shallow copy of these elements is performed.

Return Value

true if all the elements were successfully added to the vector; false otherwise.

Example

The following example adds all the elements of a LinkedList object into a Vector object at a specified index.

```
// vector_addall_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a LinkedList and add three elements to it.
        LinkedList ll = new LinkedList();
        ll.add("Linked List Element A");
        ll.add("Linked List Element B");
        ll.add("Linked List Element C");

        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("Vector Element 1");
        v.add("Vector Element 2");
        v.add("Vector Element 3");

        // Add the contents of the LinkedList to the Vector
        // at index 1 (after element "Vector Element 1").
        v.addAll(1, ll);

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
```

```
Output:
Vector Element 1
Linked List Element A
Linked List Element B
Linked List Element C
Vector Element 2
Vector Element 3
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.addElement Method

Adds an element to the end of the vector. The vector is automatically grown if there is insufficient room to add the element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void addElement(
    java.lang.Object o);
```

Parameters

o

The element to add to the end of the vector.

Example

The following example creates a Vector object and appends three elements to it.

```
// vector_addelement.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.addElement("One");
        v.addElement("Two");
        v.addElement("Three");

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
One
Two
Three
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.capacity Method

Returns the number of elements capable of being stored in the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final int capacity();
```

Return Value

The number of elements capable of being stored in the vector.

Example

The following example creates a Vector object, appends three elements to it, and then displays the capacity.

```
// vector_capacity.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("Dog");
        v.add("Cat");
        v.add("Monkey");

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }

        // Display the capacity of the Vector.
        // Note that this is not the size of the Vector.
        System.out.println("capacity = " + v.capacity());
    }
}

/*
Output:
Dog
Cat
Monkey
capacity = 10
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.clear Method

Removes all elements from the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

Example

The following example shows how to clear the contents of a Vector object.

```
// vector_clear.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("A");
        v.add("B");
        v.add("C");

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }

        // Clear the Vector and display its size.
        v.clear();
        System.out.println("size = " + v.size());
    }
}

/*
Output:
A
B
C
size = 0
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.clone Method

Creates a new instance of a Vector object that is a shallow copy of the existing vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

Return Value

A new instance of a Vector object that is a shallow copy of the existing vector.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.contains Method

Determines whether the vector contains a given element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean contains(
    java.lang.Object elem);
```

Parameters

elem

The element to search the vector for.

Return Value

The index of the element in the vector, or -1 if the element is not contained in the vector.

Example

The following example shows how to determine if a Vector contains a given element.

```
// vector_contains.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();

        Date d1 = new Date(105, Calendar.JANUARY, 1);
        Date d2 = new Date();
        Date d3 = new Date(105, Calendar.APRIL, 15);
        Date d4 = new Date(99, Calendar.DECEMBER, 31);

        v.add(d1);
        v.add(d2);
        v.add(d3);

        // Determine which of the four Date objects exist in
        // the Vector.
        boolean containsD1 = v.contains(d1);
        boolean containsD2 = v.contains(d2);
        boolean containsD3 = v.contains(d3);
        boolean containsD4 = v.contains(d4);

        System.out.println("v contains d1? " + containsD1);
        System.out.println("v contains d2? " + containsD2);
        System.out.println("v contains d3? " + containsD3);
        System.out.println("v contains d4? " + containsD4);
    }
}

/*
Output:
v contains d1? true
v contains d2? true
v contains d3? true
v contains d4? false
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.containsAll Method

Determines whether all the elements in the given collection are contained in the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean containsAll(
    java.util.Collection c);
```

Parameters

c

The collection containing the elements to search the vector for.

Return Value

true if all the elements in the collection are contained in the vector; false otherwise.

Example

The following example shows how to determine if a Vector contains all the elements of two LinkedList objects.

```
// vector_containsall.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create two LinkedList objects and a Vector object
        // containing random Dates.
        Date d1 = new Date(105, Calendar.JANUARY, 1);
        Date d2 = new Date();
        Date d3 = new Date(105, Calendar.APRIL, 15);
        Date d4 = new Date(99, Calendar.DECEMBER, 31);

        LinkedList list1 = new LinkedList();
        list1.add(d1);
        list1.add(d2);
        list1.add(d3);
        list1.add(d4);

        LinkedList list2 = new LinkedList();
        list2.add(d1);

        Vector v = new Vector();
        v.add(d1);
        v.add(d2);
        v.add(d3);

        // Determine whether the Vector contians all the objects
        // in the two LinkedList objects.
        boolean containsAllList1 = v.containsAll(list1);
        boolean containsAllList2 = v.containsAll(list2);

        System.out.println("v contains all list1? " +
            containsAllList1);
        System.out.println("v contains all list2? " +
           containsAllList2);
    }
}
```

```
/*
Output:
v contains all list1? false
v contains all list2? true
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.copyInto Method

Copies all the elements in the vector into an array.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void copyInto(
    java.lang.Object[] objArray);
```

Parameters

objArray

An array to store the elements of the vector. The array must be greater than or equal to the size of the vector. Only a shallow copy of the elements is performed.

Example

The following example shows how to copy all the contents of a Vector object into a static array.

```
// vector_copyinto.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add(new SimpleTimeZone(-28800000, "PST"));
        v.add(new SimpleTimeZone(-14400000, "EST"));
        v.add(new SimpleTimeZone(0, "GMT"));

        // Create a static array to contain the elements of
        // the Vector.
        TimeZone[] tzArr = new TimeZone[3];
        v.copyInto(tzArr);

        // Display the contents of the static array.
        for (int i = 0; i < tzArr.length; i++)
        {
            System.out.println(tzArr[i].getID());
        }
    }
}

/*
Output:
PST
EST
GMT
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.elementAt Method

Returns the element at the specified index. The element is not removed from the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized java.lang.Object elementAt(
    int index);
```

Parameters

index

The index of the element to be retrieved. This value must be less than the overall length of the vector.

Return Value

The element present at the specified index of the vector.

Example

The following example shows how to retrieve the element at a given index in the Vector object.

```
// vector_elementat.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("One");
        v.add("Two");
        v.add("Three");

        // Display the contents of the Vector.
        for (int i = 0; i < v.size(); i++)
        {
            System.out.println(v.elementAt(i));
        }
    }
}

/*
Output:
One
Two
Three
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.elements Method

Returns an Enumeration object that can be used to efficiently traverse the elements in the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized java.util.Enumeration elements();
```

Return Value

An Enumeration object that can be used to efficiently traverse the elements in the vector.

Example

The following example shows how to traverse a Vector object using an Enumeration.

```
// vector_elements.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        // Order should be "Orange", "Banana", "Apple".
        Vector v = new Vector();
        v.add(0, "Apple");
        v.add(0, "Orange");
        v.add(1, "Banana");

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
Orange
Banana
Apple
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.ensureCapacity Method

Determines whether the vector can hold the given number of elements. If it cannot, a new vector is created that is large enough to store the desired number of elements and the contents of the old vector are copied into the newly created vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void ensureCapacity(
    int minCapacity);
```

Parameters

minCapacity

The desired number of elements to store in the vector.

Example

The following example shows one way to manually grow a Vector object once its capacity is reached using ensureCapacity.

```
// vector_ensurecapacity.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector(3);
        v.add(new SimpleTimeZone(-28800000, "PST"));
        v.add(new SimpleTimeZone(-14400000, "EST"));
        v.add(new SimpleTimeZone(0, "GMT"));

        // Grow the Vector if it is full.
        if (v.capacity() == v.size())
        {
            v.ensureCapacity(v.size() * 2);
        }

        // Display the capacity of the Vector.
        System.out.println("new capacity = " + v.capacity());
    }
}

/*
Output:
new capacity = 6
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.equals Method

Determines whether two vectors are equal. Two vectors are considered equal if they contain exactly the same elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean equals(
    java.lang.Object obj);
```

Parameters

obj

The collection to compare for equality with the vector.

Return Value

true if the provided collection contains exactly the same elements as the vector; false otherwise.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.firstElement Method

Returns the first element in the vector. The element is not removed from the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized java.lang.Object firstElement();
```

Return Value

The first element in the vector.

Example

The following example shows how to retrieve the first element in a Vector.

```
// vector_firstelement.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("First Element");
        v.add("Second Element");
        v.add("Third Element");

        // Display the first element of the Vector.
        System.out.println(v.firstElement());
    }
}

/*
Output:
First Element
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.get Method

Returns the element at a given index. The element is not removed from the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.Object get(
    int ix);
```

Parameters

ix

The index of the element to retrieve. This value must be less than the number of elements in the vector.

Return Value

The element at the given index in the vector.

Example

The following example shows how to retrieve the element at a given index in the Vector object.

```
// vector_get.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("One");
        v.add("Two");
        v.add("Three");

        // Display the contents of the Vector.
        for (int i = 0; i < v.size(); i++)
        {
            System.out.println(v.get(i));
        }
    }
}

/*
Output:
One
Two
Three
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.hashCode Method

Provides a hash value representing this vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized int hashCode();
```

Return Value

A hash value representing this vector.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.indexOf Method

Returns the index of the first occurrence of a given element.

Overload List

Name	Description
Vector.indexOf (Object)	Returns the index of the first occurrence of a given element.
Vector.indexOf (Object, int)	Returns the index of the first occurrence of a given element. The vector is searched starting from the given index.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.indexOf Method (Object)

Returns the index of the first occurrence of a given element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int indexOf(
    java.lang.Object obj);
```

Parameters

obj

The element to search the vector for.

Return Value

The index of the first occurrence of the given element.

Example

The following example shows how to determine the index of an object in a Vector.

```
// vector_indexof.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();

        Date d1 = new Date(105, Calendar.JANUARY, 1);
        Date d2 = new Date();
        Date d3 = new Date(105, Calendar.APRIL, 15);
        Date d4 = new Date(99, Calendar.DECEMBER, 31);

        v.add(d1);
        v.add(d2);
        v.add(d3);

        // Determine the indices of the various Date objects
        // in the Vector.
        int indexD1 = v.indexOf(d1);
        int indexD2 = v.indexOf(d2);
        int indexD3 = v.indexOf(d3);
        int indexD4 = v.indexOf(d4);

        System.out.println("v contains d1 at index " + indexD1);
        System.out.println("v contains d2 at index " + indexD2);
        System.out.println("v contains d3 at index " + indexD3);
        System.out.println("v contains d4 at index " + indexD4);
    }
}

/*
Output:
v contains d1 at index 0
v contains d2 at index 1
v contains d3 at index 2
v contains d4 at index -1
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.indexOf Method (Object, Int32)

Returns the index of the first occurrence of a given element. The vector is searched starting from the given index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized int indexOf(
    java.lang.Object obj,
    int index);
```

Parameters

obj

The element to search the vector for.

index

Represents the index of the element in the vector to start the search. This value must be less than the number of elements in the vector.

Return Value

The index of the first occurrence of the given element after the starting index.

Example

The following example shows how to determine all the indices of an object in a Vector.

```
// vector_indexof_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();

        Date d1 = new Date(105, Calendar.JANUARY, 1);
        Date d2 = new Date();
        Date d3 = new Date(105, Calendar.APRIL, 15);
        Date d4 = new Date(99, Calendar.DECEMBER, 31);

        v.add(d1);
        v.add(d2);
        v.add(d3);
        v.add(d1);
        v.add(d3);

        // Determine all the indices of a Date object
        // in the Vector.
        int currIndex = 0;
        while (true)
        {
            currIndex = v.indexOf(d1, currIndex);
            if (currIndex == -1)
                break;
            else
            {
                System.out.println("v contains d1 at index " + currIndex);
                currIndex++;
            }
        }
```

```
    }
}

/*
Output:
v contains d1 at index 0
v contains d1 at index 3
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.insertElementAt Method

Inserts an element into the vector at the given index. The vector is automatically grown if there is insufficient room to add the element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void insertElementAt(
    java.lang.Object obj,
    int index);
```

Parameters

obj

The element to insert into the vector.

index

The index to insert the element into. This value must be less than the number of elements in the vector.

Example

The following example creates a Vector object and adds three elements to it at the specified indices.

```
// vector_insertelementat.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        // Order should be "Orange", "Banana", "Apple".
        Vector v = new Vector();
        v.insertElementAt("Apple", 0);
        v.insertElementAt("Orange", 0);
        v.insertElementAt("Banana", 1);

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
Orange
Banana
Apple
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.isEmpty Method

Determines whether the vector contains any elements.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final boolean isEmpty();
```

Return Value

true if the vector contains no elements; false otherwise.

Example

The following example checks to see if a Vector object is empty before enumerating over its contents.

```
// vector_isempty.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("1");
        v.add("2");
        v.add("3");

        // Display the contents of the Vector if it is not empty.
        if (!v.isEmpty())
        {
            Enumeration vEnum = v.elements();
            while (vEnum.hasMoreElements())
            {
                System.out.println(vEnum.nextElement());
            }
        }
    }
}

/*
Output:
1
2
3
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.lastElement Method

Returns the last element in the vector. The element is not removed from the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized java.lang.Object lastElement();
```

Return Value

The last element in the vector.

Example

The following example shows how to retrieve the last element in a Vector.

```
// vector_lastelement.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("First Element");
        v.add("Second Element");
        v.add("Third Element");

        // Display the last element of the Vector.
        System.out.println(v.lastElement());
    }
}

/*
Output:
Third Element
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.lastIndexOf Method

Returns the index of the last occurrence of a given element.

Overload List

Name	Description
Vector.lastIndexOf (Object)	Returns the index of the last occurrence of a given element.
Vector.lastIndexOf (Object, int)	Returns the index of the last occurrence of a given element. The vector is searched starting fro m the beginning until the given index.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.lastIndexOf Method (Object)

Returns the index of the last occurrence of a given element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int lastIndexOf(
    java.lang.Object obj);
```

Parameters

obj

The element to search the vector for.

Return Value

The index of the last occurrence of the given element.

Example

The following example shows how to determine the last index of an object in a Vector.

```
// vector_lastindexof.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add six elements to it.
        Vector v = new Vector();

        Date d1 = new Date(105, Calendar.JANUARY, 1);
        Date d2 = new Date();
        Date d3 = new Date(105, Calendar.APRIL, 15);
        Date d4 = new Date(99, Calendar.DECEMBER, 31);

        v.add(d1);
        v.add(d2);
        v.add(d3);
        v.add(d3);
        v.add(d2);
        v.add(d1);

        // Determine the last index of the various Date objects
        // in the Vector.
        int indexD1 = v.lastIndexOf(d1);
        int indexD2 = v.lastIndexOf(d2);
        int indexD3 = v.lastIndexOf(d3);
        int indexD4 = v.lastIndexOf(d4);

        System.out.println("v contains d1 at last index of " +
            indexD1);
        System.out.println("v contains d2 at last index of " +
            indexD2);
        System.out.println("v contains d3 at last index of " +
            indexD3);
        System.out.println("v contains d4 at last index of " +
            indexD4);
    }
}
```

```
/*
Output:
v contains d1 at last index of 5
v contains d2 at last index of 4
v contains d3 at last index of 3
v contains d4 at last index of -1
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.lastIndexOf Method (Object, Int32)

Returns the index of the last occurrence of a given element. The vector is searched starting from the beginning until the given index.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized int lastIndexOf(
    java.lang.Object obj,
    int index);
```

Parameters

obj

The element to search the vector for.

index

Represents the index of the element in the vector to end the search. The search is started from the beginning of the vector. This value must be less than the number of elements in the vector.

Return Value

The index of the last occurrence of the given element before the ending index.

Example

The following example shows how to determine all the indices of an object in a Vector in reverse order.

```
// vector_lastindexof_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add five elements to it.
        Vector v = new Vector();

        Date d1 = new Date(105, Calendar.JANUARY, 1);
        Date d2 = new Date();
        Date d3 = new Date(105, Calendar.APRIL, 15);
        Date d4 = new Date(99, Calendar.DECEMBER, 31);

        v.add(d1);
        v.add(d2);
        v.add(d3);
        v.add(d1);
        v.add(d3);

        // Determine all the indices of a Date object
        // in the Vector.
        int currIndex = v.size() - 1;
        while (true)
        {
            currIndex = v.lastIndexOf(d1, currIndex);
            if (currIndex == -1)
                break;
            else
            {
                System.out.println("v contains d1 at index " + currIndex);
                currIndex--;
```

```
            }
        }
    }
}

/*
Output:
v contains d1 at index 3
v contains d1 at index 0
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.remove Method

Removes an element from the vector.

Overload List

Name	Description
Vector.remove (int)	Removes the element from the vector at the specified index. The elements in the vector beyond this index are then shifted down.
Vector.remove (Object)	Removes the first occurrence of an element from the vector. The elements in the vector beyond this element are then shifted down.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.remove Method (Int32)

Removes the element from the vector at the specified index. The elements in the vector beyond this index are then shifted down.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object remove(
    int ix);
```

Parameters

ix

The index of the element to be removed from the vector. This value must be less than the number of elements in the vector.

Return Value

The element at the specified index.

Example

The following example shows how to remove an element at a given index from a Vector.

```
// vector_remove.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("One");
        v.add("Two");
        v.add("Three");

        // Remove the element at index 1.
        v.remove(1);

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
One
Three
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.remove Method (Object)

Removes the first occurrence of an element from the vector. The elements in the vector beyond this element are then shifted down.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean remove(
    java.lang.Object e);
```

Parameters

e

The element to be removed from the vector.

Return Value

true if the element was successfully removed; false otherwise.

Example

The following example shows how to remove a given element from a Vector.

```
// vector_remove_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();

        Date d1 = new Date(105, Calendar.JANUARY, 1);
        Date d2 = new Date();
        Date d3 = new Date(105, Calendar.APRIL, 15);
        Date d4 = new Date(99, Calendar.DECEMBER, 31);

        v.add(d1);
        v.add(d2);
        v.add(d3);

        // Remove d1 and d4 from the Vector.
        boolean d1Removed = v.remove(d1);
        boolean d4Removed = v.remove(d4);

        // Display the results of the remove operation.
        System.out.println("d1 successfully removed? " + d1Removed);
        System.out.println("d4 successfully removed? " + d4Removed);
    }
}

/*
Output:
d1 successfully removed? true
d4 successfully removed? false
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.removeAll Method

Removes all the elements contained in a collection from the vector. If an element appears more than once in the vector, all instances are removed. The elements in the vector beyond these elements are then shifted down.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean removeAll(
    java.util.Collection c);
```

Parameters

c

A collection of elements to be removed from the vector.

Return Value

true if at least one element in the collection was removed from the vector; false otherwise.

Example

The following example shows how to remove all elements in a LinkedList from a Vector.

```
// vector_removeall.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a LinkedList and add three elements to it.
        LinkedList ll = new LinkedList();
        ll.add("Linked List Element A");
        ll.add("Linked List Element B");
        ll.add("Linked List Element C");

        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("Vector Element 1");
        v.add("Vector Element 2");
        v.add("Vector Element 3");

        // Add the contents of the LinkedList to the Vector
        // at index 1 (after element "Vector Element 1").
        v.addAll(1, ll);

        // Now remove all the elements in the LinkedList from
        // the Vector.
        v.removeAll(ll);

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
Vector Element 1
```

```
Vector Element 2
Vector Element 3
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.removeAllElements Method

Removes all elements from the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void removeAllElements();
```

Example

The following example shows how to clear the contents of a Vector object.

```
// vector_removeallelements.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("A");
        v.add("B");
        v.add("C");

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }

        // Clear the Vector and display its size.
        v.removeAllElements();
        System.out.println("size = " + v.size());
    }
}

/*
Output:
A
B
C
size = 0
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.removeElement Method

Removes the first occurrence of an element from the vector. The elements in the vector beyond this element are then shifted down.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized boolean removeElement(
    java.lang.Object obj);
```

Parameters

obj

The element to be removed from the vector.

Return Value

true if the element was successfully removed from the vector; false otherwise.

Example

The following example shows how to remove a given element from a Vector.

```
// vector_removeelement.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();

        Date d1 = new Date(105, Calendar.JANUARY, 1);
        Date d2 = new Date();
        Date d3 = new Date(105, Calendar.APRIL, 15);
        Date d4 = new Date(99, Calendar.DECEMBER, 31);

        v.add(d1);
        v.add(d2);
        v.add(d3);

        // Remove d1 and d4 from the Vector.
        boolean d1Removed = v.removeElement(d1);
        boolean d4Removed = v.removeElement(d4);

        // Display the results of the remove operation.
        System.out.println("d1 successfully removed? " + d1Removed);
        System.out.println("d4 successfully removed? " + d4Removed);
    }
}

/*
Output:
d1 successfully removed? true
d4 successfully removed? false
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.removeElementAt Method

Removes the element from the vector at the specified index. The elements in the vector beyond this index are then shifted down.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void removeElementAt(
    int index);
```

Parameters

index

The index of the element to be removed from the vector. This value must be less than the number of elements in the vector.

Example

The following example shows how to remove an element at a given index from a Vector.

```
// vector_removeelementat.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add("One");
        v.add("Two");
        v.add("Three");

        // Remove the element at index 1.
        v.removeElementAt(1);

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
One
Three
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.removeRange Method

Removes all the elements from the vector starting with the element at fromIx to the element at one less than toIx.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
protected void removeRange(
    int fromIx,
    int toIx);
```

Parameters

fromIx

The index of the first element to be removed from the vector. This value must be less than the number of elements in the vector.

toIx

The index one greater than the last element to be removed from the vector. This value must be greater than fromIx and less than the number of elements in the vector.

See Also
Reference
Vector Class
Concepts
Vector Members
java.util Package

Vector.retainAll Method

Removes all elements from the vector that are not contained in the given collection.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized boolean retainAll(
    java.util.Collection c);
```

Parameters

c

The collection of elements that are to be retained in the vector. All elements not in this collection are removed from the vector.

Return Value

true if at least one element was removed from the vector; false otherwise.

Example

The following example shows how to retain all the elements in a Vector that are contained in a LinkedList, and to dispose of all other elements in the Vector.

```
// vector_retainall.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a LinkedList and add three elements to it.
        LinkedList ll = new LinkedList();
        ll.add("A");
        ll.add("B");
        ll.add("C");

        // Create a Vector containing the same elements as the
        // LinkedList and more.
        Vector v = new Vector();
        v.add("A");
        v.add("B");
        v.add("C");
        v.add("D");
        v.add("E");

        // Retain only those elements found in the LinkedList.
        v.retainAll(ll);

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
A
B
```

```
C
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.set Method

Sets the element at the given index to the given element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.Object set(
    int ix,
    java.lang.Object e);
```

Parameters

ix

The index in the vector where the element is to be set.

e

The element to set at the given index in the vector.

Return Value

The element that was at the given index before the element was changed.

Example

The following example shows how to set an element at a given index in the Vector.

```
// vector_set.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add some elements to it.
        Vector v = new Vector();
        v.add("One");
        v.add("Two");
        v.add("Three");

        // Change the element at index 1.
        Object oldTwo = v.set(1, "Four");

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
One
Four
Three
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.setElementAt Method

Sets the element at the given index to the given element.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void setElementAt(
    java.lang.Object obj,
    int index);
```

Parameters

obj

The element to set at the given index in the vector.

index

The index in the vector where the element is to be set.

Example

The following example shows how to set an element at a given index in the Vector.

```
// vector_setelementat.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add some elements to it.
        Vector v = new Vector();
        v.add("1");
        v.add("2");
        v.add("3");

        // Change the element at index 1.
        v.setElementAt("4", 1);

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
1
4
3
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.setSize Method

Sets the size of the vector to the given value. If the given size is greater than the current size of the vector, then the contents of the old vector are copied into a new vector of the given size. If the given size is less than the current size of the vector, then some elements will be removed from the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void setSize(
    int newSize);
```

Parameters

newSize

The new size of the vector.

Example

The following example shows how to shrink a Vector object by decreasing its size.

```
// vector_setsize.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add some elements to it.
        Vector v = new Vector();
        v.add("Dog");
        v.add("Cat");
        v.add("Bird");

        // Change the size of the Vector.
        v.setSize(2);

        // Display the contents of the Vector.
        Enumeration vEnum = v.elements();
        while (vEnum.hasMoreElements())
        {
            System.out.println(vEnum.nextElement());
        }
    }
}

/*
Output:
Dog
Cat
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.size Method

Returns the number of elements in the vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final int size();
```

Return Value

The number of elements in the vector.

Example

The following example shows how to determine the size of a Vector.

```
// vector_size.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a LinkedList and add three elements to it.
        LinkedList ll = new LinkedList();
        ll.add("A");
        ll.add("B");
        ll.add("C");

        // Create a Vector containing the same elements as the
        // LinkedList and more.
        Vector v = new Vector();
        v.add("A");
        v.add("B");
        v.add("C");
        v.add("D");
        v.add("E");

        // Retain only those elements found in the LinkedList.
        v.retainAll(ll);

        // Display the size of the Vector.
        System.out.println("size = " + v.size());
    }
}

/*
Output:
size = 3
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.subList Method

Returns a List of all the elements in the vector starting with the element at fromIx to the element at one less than toIx.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.List subList(
    int fromIx,
    int toIx);
```

Parameters

fromIx

The index of the first element in the vector to be inserted into the List. This value must be less than the number of elements in the vector.

toIx

The index one greater than the last element in the vector to be inserted into the List. This value must be greater than fromIx and less than the number of elements in the vector.

Return Value

A List object containing the elements in the given range.

Example

The following example shows hw to return a sub-list of the elements in a Vector.

```
// vector_sublist.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add four elements to it.
        Vector v = new Vector();

        Date d1 = new Date(105, Calendar.JANUARY, 1);
        Date d2 = new Date();
        Date d3 = new Date(105, Calendar.APRIL, 15);
        Date d4 = new Date(99, Calendar.DECEMBER, 31);

        v.add(d1);
        v.add(d2);
        v.add(d3);
        v.add(d4);

        // Get a sublist of the elements in the range [1, 3).
        List l = v.subList(1, 3);

        // Display the elements in the List.
        Iterator iter = l.iterator();
        while (iter.hasNext())
        {
            System.out.println(iter.next());
        }
    }
}

/*
```

```
Output:
Thu Sep 09 18:08:03 PDT 2004
Fri Apr 15 00:00:00 PDT 2005
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.toArray Method

Converts a vector into a static array.

Overload List

Name	Description
Vector.toArray ()	Converts a vector into a static array.
Vector.toArray (Object[])	Converts a vector into a static array.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.toArray Method ()

Converts a vector into a static array.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.Object[] toArray();
```

Return Value

A static array containing the elements of the vector. This array represents a shallow copy of the original vector.

Example

The following example shows how to populate a static array with the contents of a Vector.

```
// vector_toarray.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add(new SimpleTimeZone(-28800000, "PST"));
        v.add(new SimpleTimeZone(-14400000, "EST"));
        v.add(new SimpleTimeZone(0, "GMT"));

        // Create a static array to contain the elements of
        // the Vector.
        Object[] tzArr = new TimeZone[v.size()];
        tzArr = v.toArray();

        // Display the contents of the static array.
        for (int i = 0; i < tzArr.length; i++)
        {
            System.out.println(((SimpleTimeZone)tzArr[i]).getID());
        }
    }
}

/*
Output:
PST
EST
GMT
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.toArray Method (Object[ ])

Converts a vector into a static array.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public synchronized java.lang.Object[] toArray(
    java.lang.Object[] arr);
```

Parameters

arr

The array to add the elements of the vector to.

Return Value

A static array containing the elements of the vector. This array represents a shallow copy of the original vector.

Example

The following example shows how to populate a static array with the contents of a Vector.

```
// vector_toarray_2.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector();
        v.add(new SimpleTimeZone(-28800000, "PST"));
        v.add(new SimpleTimeZone(-14400000, "EST"));
        v.add(new SimpleTimeZone(0, "GMT"));

        // Create a static array to contain the elements of
        // the Vector.
        Object[] tzArr = new TimeZone[v.size()];
        tzArr = v.toArray(tzArr);

        // Display the contents of the static array.
        for (int i = 0; i < tzArr.length; i++)
        {
            System.out.println(((SimpleTimeZone)tzArr[i]).getID());
        }
    }
}

/*
Output:
PST
EST
GMT
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.toString Method

Displays a human-readable representation of a vector.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized java.lang.String toString();
```

Return Value

A human-readable representation of a vector.

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

Vector.trimToSize Method

Changes the capacity of a vector to match its size.

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public final synchronized void trimToSize();
```

Example

The following example shows how to trim a Vector to the number of elements it contains.

```
// vector_trimtosize.jsl

import java.util.*;

public class Program
{
    public static void main(String[] args)
    {
        // Create a Vector and add three elements to it.
        Vector v = new Vector(20);
        v.add("One");
        v.add("Two");
        v.add("Three");

        // Trim the Vector from a capacity of 20 to 3.
        v.trimToSize();

        // Display the capacity of the Vector.
        System.out.println("capacity = " + v.capacity());
    }
}

/*
Output:
capacity = 3
*/
```

See Also

Reference

Vector Class

Concepts

Vector Members

java.util Package

WeakHashMap Class

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public class java.util.WeakHashMap
    extends java.util.AbstractMap
    implements java.util.Map
```

Inheritance Hierarchy

java.lang.Object

  java.util.AbstractMap

   java.util.WeakHashMap

See Also

Concepts

WeakHashMap Members

java.util Package

WeakHashMap Members

The following tables list the members exposed by the WeakHashMap type.

Public Constructors

Name	Description
WeakHashMap	Overloaded.

Public Methods

Name	Description
clear	Overridden.
containsKey	Overridden.
containsValue	(inherited from AbstractMap)
entrySet	Overridden.
equals	(inherited from AbstractMap)
get	Overridden.
hashCode	(inherited from AbstractMap)
getClass	(inherited from Object)
isEmpty	Overridden.
keySet	(inherited from AbstractMap)
clone	(inherited from AbstractMap)
put	Overridden.
putAll	(inherited from AbstractMap)
remove	Overridden.
size	Overridden.
toString	(inherited from AbstractMap)
values	(inherited from AbstractMap)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

WeakHashMap Class

Concepts

java.util Package

WeakHashMap Constructor

Overload List

Name	Description
WeakHashMap ()	
WeakHashMap (int)	
WeakHashMap (int, float)	

See Also

Reference

WeakHashMap Class

Concepts

WeakHashMap Members

java.util Package

WeakHashMap Constructor ()

Initializes a new instance of the WeakHashMap Class .

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.WeakHashMap();
```

Remarks

The default constructor initializes any fields to their default values.

See Also

Reference

WeakHashMap Class

Concepts

WeakHashMap Members

java.util Package

WeakHashMap Constructor (Int32)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.WeakHashMap(
    int initCap);
```

Parameters
initCap

See Also
Reference
WeakHashMap Class
Concepts
WeakHashMap Members
java.util Package

WeakHashMap Constructor (Int32, Single)

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.WeakHashMap(
    int initCap,
    float loadFactor);
```

Parameters

initCap

loadFactor

See Also
Reference
WeakHashMap Class
Concepts
WeakHashMap Members
java.util Package

WeakHashMap Methods

Public Methods

Name	Description
clear	Overridden.
containsKey	Overridden.
containsValue	(inherited from AbstractMap)
entrySet	Overridden.
equals	(inherited from AbstractMap)
get	Overridden.
hashCode	(inherited from AbstractMap)
getClass	(inherited from Object)
isEmpty	Overridden.
keySet	(inherited from AbstractMap)
clone	(inherited from AbstractMap)
put	Overridden.
putAll	(inherited from AbstractMap)
remove	Overridden.
size	Overridden.
toString	(inherited from AbstractMap)
values	(inherited from AbstractMap)

Protected Methods

Name	Description
finalize	(inherited from Object)

See Also

Reference

WeakHashMap Class

Concepts

java.util Package

WeakHashMap.clear Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public void clear();
```

See Also

Reference

WeakHashMap Class

Concepts

WeakHashMap Members

java.util Package

WeakHashMap.containsKey Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean containsKey(
    java.lang.Object k);
```

Parameters
k

See Also
Reference
WeakHashMap Class
Concepts
WeakHashMap Members
java.util Package

WeakHashMap.entrySet Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.util.Set entrySet();
```

See Also

Reference

WeakHashMap Class

Concepts

WeakHashMap Members

java.util Package

WeakHashMap.get Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object get(
    java.lang.Object k);
```

Parameters
k

See Also
Reference
WeakHashMap Class
Concepts
WeakHashMap Members
java.util Package

WeakHashMap.isEmpty Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public boolean isEmpty();
```

See Also

Reference

WeakHashMap Class

Concepts

WeakHashMap Members

java.util Package

WeakHashMap.put Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object put(
    java.lang.Object k,
    java.lang.Object v);
```

Parameters

k

v

See Also
Reference
WeakHashMap Class
Concepts
WeakHashMap Members
java.util Package

WeakHashMap.remove Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public java.lang.Object remove(
    java.lang.Object k);
```

Parameters
k

See Also
Reference
WeakHashMap Class
Concepts
WeakHashMap Members
java.util Package

WeakHashMap.size Method

Package: java.util

Assembly: vjslib (in vjslib.dll)

```
public int size();
```

See Also

Reference

WeakHashMap Class

Concepts

WeakHashMap Members

java.util Package

Visual J# のサンプル

ここでは、Visual J# のサンプルの要約を示します。各トピックには、サンプルのソース ファイルを開いたりコピーしたりするためのリンクが含まれています。どのサンプルにもソリューション ファイル (.sln) が付属しているので、Visual Studio 開発環境でサンプルを開き、ビルドしてデバッグできます。

このセクションの内容

アプリケーション サンプル

完成済みの一般的なアプリケーションにならって作成されたサンプルです。

複数言語のサンプル

他の言語で開発されたコンポーネントを使うサンプルです。

技術サンプル

プラットフォーム呼び出し、リソースの読み込み、セキュリティ、属性、イベント、デリゲートなど、個々のテクノロジおよび言語機能を示すサンプルです。

Visual J# アプリケーションの高度なサンプル

完成済みの一般的なアプリケーションにならって作成されたサンプルです。「Visual J# アプリケーションのサンプル」で紹介されているサンプルよりも難易度の高いアプリケーション サンプルが紹介されています。

Visual J# の複数の言語を含む高度なサンプル

他の言語で記述されたサンプルから移植されるサンプルへのリンクを示します。

Visual J# の高度な技術サンプル

.NET Framework および Visual J# のさまざまなテクノロジと機能を説明するサンプルを示します。各サンプルは、1 つのテクノロジまたは関連するいくつかのテクノロジについて説明することを目的としています。

関連するセクション

.NET Framework のサンプル

.NET Framework と .NET Compact Framework のさまざまな機能について説明します。

Visual J# アプリケーションのサンプル

このセクションのサンプルは、完成済みの一般的なアプリケーションにならって作成されています。これらのサンプルには、Visual J# の他のサンプルと比較して、より完全なエラー処理とオブジェクト指向開発の例が示されています。

サンプル	説明
Dice	Visual J# で C# クラスを利用する方法を示します。
TypeFinder	ユーザーの環境から型に関する情報を取得するためのコマンド ライン インターフェイスを提供します。また、リフレクションについての一般情報も提供します。
WinTalk	Windows フォーム、ソケット、およびその他の FCL のトピックについて、概要を簡単に示します。このサンプルは、非常に単純なソケット チャット アプリケーションです。
WordCount	.NET Framework クラス ライブラリの概要を示します。テキスト ファイル内の単語数をカウントし、その結果をコマンド ライン引数に応じて編成するアプリケーションの例を示します。

関連するセクション

Visual J# のサンプル

Visual J# の技術サンプル

Visual J# の複数の言語を含むサンプル

Dice サンプル (C# クラスの利用)

Download sample

このサンプルでは、次の手段によって Visual J# から C# のクラスを利用する方法を示します。

- C# で定義されたインターフェイスを Visual J# クラスに実装する。
- C# クラスからスローされた例外を Visual J# で処理する。
- C# クラスによって公開されるプロパティを Visual J# で検査する。
- C# クラスから発生したイベントを Visual J# でシンクする。

cscServer プロジェクトは、ゲームで使用されるようなさいころを実装します。

さいころの面 (表面) の数は、プログラムで設定できます。さいころは、**Roll** メソッドによって投げられます。**Roll** 操作では、さいころの上になった面の数値が返されます。

さいころを投げる操作では、**IRoll** インターフェイスが使用されます。クライアントは、このインターフェイスの実装を用意する必要があります。クライアントは、さいころの **Roll** メソッドを呼び出す前に、このインターフェイスをさいころに設定する必要があります。

さいころが無効な面の数で構築された場合 (たとえば、面の数が 0 以下の場合) は、**BadDieSizeException** 例外がスローされます。面の数を設定する前にさいころの **Roll** メソッドを呼び出すと、NullReferenceException 例外がスローされます。

さいころは、**NumSides** という読み取り専用プロパティを公開します。このプロパティは、さいころの面の数を調べるために使用できます。

さいころに **IRoll** インターフェイスの実装が設定されるたびに、**LoadedDice** イベントが発生します。

Dice プロジェクトでは、次の処理を行います。

- **IRoll** インターフェイスの実装を提供する。
- スローされた例外を両方ともキャッチする。
- **NumSides** プロパティを調べる。
- さいころから発生したイベントをシンクするために、デリゲート オブジェクトを用意する。

セキュリティに関するメモ :
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

開発環境でビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- 開発環境では、F5 キーを押します。

  または

  コマンド ラインでは、「Dice」と入力し、**Enter** キーを押します。

参照

その他の技術情報

Visual J# アプリケーションのサンプル

TypeFinder サンプル (型情報の取得)

Download sample

TypeFinder は、環境の型に関する情報を取得するためのコマンド ライン インターフェイスを提供します。特定の型の名前空間や DLL を見つけることは困難な場合があります。型が見つかっても、ドキュメントがなかったり、ドキュメントが古かったりすることもあります。このユーティリティを使うと、使用できる型、その型が格納されているモジュール、その型で使用できるインターフェイス、メソッド、フィールド、プロパティ、およびイベントを簡単に確認できます。このサンプルでは、リフレクションの概要についても説明します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

開発環境でサンプルをビルドして実行するには

1. [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。
2. F5 キーを押して、サンプルを実行します。

コマンド ラインでサンプルをビルドして実行するには

1. 「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

2. 「**TypeFinder**」と入力し、Enter キーを押します。
3. ある単語を名前に含む各型の場所を検索するには、コマンド ラインで「**TypeFinder** *word* 」と入力します。

サンプルのクラスとテクノロジ

このサンプルでは、次のクラスとテクノロジが使用されています。

リフレクション

クラス/テクノロジ	説明
Assembly	アセンブリを AppDomain に読み込んで、型を検索できるようにします。

Module	検索文字列と比較するために、アセンブリまたはモジュールから型を取得します。
Type	名前、名前空間、メンバなどの型情報を取得します。
PropertyInfo	型のプロパティに関する情報を検索します。
EventInfo	型のイベントに関する情報を検索します。
FieldInfo	型のフィールドに関する情報を検索します。
MethodInfo	型のメソッドに関する情報を検索します。

入出力

クラス/テクノロジ	説明
TextWriter	汎用的な方法で出力するために、サンプルの `IndentedWriter` 型によって使用されます。既定では、出力先はコンソールになります。

テキスト

クラス/テクノロジ	説明
StringBuilder	文字列を作成するために、サンプルの `IndentedWriter` 型によって使用されます。
String	書式指定文字列、部分文字列、大文字の文字列などを検索するために、サンプル全体をとおして使用されます。

コレクション

クラス/テクノロジ	説明
ArrayList	文字列のリストを管理するために、サンプルによって使用されます。

レジストリ

クラス/テクノロジ	説明
Registry	`LocalMachine` キーのサブキーに対して、`RegistryKey` 型のインスタンスを作成するために使います。
RegistryKey	レジストリのキーから値を読み取るために使います。

参照

その他の技術情報

Visual J# アプリケーションのサンプル

WinTalk サンプル (Windows フォームおよび Windows ソケットの使用)

Download sample

このサンプルでは、Windows フォーム、ソケット、およびその他の Framework クラス ライブラリ (FCL: Framework Class Library) のトピックについて簡単な概要を示します。このサンプルは、非常に単純なソケット チャット アプリケーションです。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

1. このユーティリティのコマンド ライン オプションの一覧を表示するには、コマンド ラインで「**WinTalk.exe /?**」と入力します。
2. クライアントとサーバーを同じコンピュータで実行してサンプルを試すには、単にアプリケーションをパラメータなしで 2 回実行します。
3. ポート番号を指定するときは、他のプログラムでそのポートが使用されていないことを確認してください。他のプログラムが使っているポートを指定すると、サンプルが正しく実行されません。上位のポート番号 5001 ～ 5150 を使ってください。
4. サンプルが正しく機能するようにするには、**Path**、**Include**、**Lib** の各環境変数を適切に設定する必要があります。これらの変数を設定するには、*<SDKRoot>\Bin* ディレクトリにある vsVars.bat を実行します。vsVars.bat は、すべてのコマンド シェルで実行する必要があります。

サンプルのクラスとテクノロジ

このサンプルでは、次のクラスとテクノロジが使用されています。

Windows フォーム

クラス/テクノロジ	説明
Application	メッセージ ポンプを実装するために使います。
Splitter	分割のセマンティクスを処理するコントロール型です。送信と受信のテキスト フレームを分割するために使います。
Panel	フォーム上のコントロールを自由に配置、ドッキング、および固定できます。このサンプルのパネルには、2 つのテキスト ボックスと分割線が含まれます。一方、ステータス用のコントロールはパネルの兄弟になります。
TextBox	サンプルの 2 つのエディット コントロールを実装します。エディット コントロールの 1 つは受信テキスト用のため、読み取り専用です。
Label	サンプルの静的なステータス情報を表示するために使います。
MessageBox	エラーやシャットダウンの状況を通知するために使います。

ネットワーキング

クラス/テクノロジ	説明
IPEndPoint	IP アドレスおよびポート番号をカプセル化するために使います。
IPAddress	**IPAddress** 関連の便利なロジックを提供します。
Dns	DNS 名から IP アドレスを取得するために使います。
Socket	マネージ コードへのネットワーク機能の実装は、多くの場合、XML Web サービスなどの抽象化された機能をとおして行われます。ただし、ソケット コードを直接実装することもできます。このサンプルでは、**Socket** クラスを使ってこの手順を示します。
NetworkStream	stream から派生し、自身の基底のデバイスとしてソケットを使います。これは、ソケット経由でデータをストリーム転送する場合に便利です。**NetworkStream** 型は、**FileStream** などの他のストリーム型と同じように使用できます。

入出力

クラス/テクノロジ	説明
StreamWriter	ソケットを表すネットワーク ストリームに書き込むために使います。
StreamReader	ソケットを表すネットワーク ストリームから読み取るために使います。

ガベージ コレクション

クラス/テクノロジ	説明
GC	既に破棄されているカスタム オブジェクトの終了処理を抑制するために使います。

デリゲート

クラス/テクノロジ	説明
MulticastDelegate	**MulticastDelegate** から派生するデリゲート型を実装します。この型は、ネットワーク イベントに関連する通知に使用されます。サンプルでは、このデリゲート型を使ってネットワーク型をフォーム型と連結します。

スレッド処理

クラス/テクノロジ	説明

ThreadPool	このサンプルのカスタム *Talker* クラスは、ネットワーク ロジックを処理し、2 つ目のスレッドを使ってネットワークから読み取ります。ただし、スレッドを作成するのではなく、共通言語ランタイムのスレッド プールのスレッドを使います。これは、マネージ コードのマルチスレッドに適した方法です。

例外

クラス/テクノロジ	説明
IOException	通常、**StreamReader** 型と **StreamWriter** 型は、**IOException** 型をスローします。ただし、これらの例外を正しくキャッチして処理するには、多くの場合、基になるエラーを示す付加された例外をチェックする必要があります。このサンプルの場合は、**SocketException** 型になります。
SocketException	このサンプルでは、**SocketException** 型をキャッチし、ソケットの **ErrorCode** プロパティで予測される特定のエラー型をチェックします。

コマンド ライン要件

Visual J# サンプルをコマンド ラインでビルドして実行する場合、サンプルが正しく機能するようにするには、Path 環境変数を適切に設定する必要があります。Path 変数は、<%windir%>\Microsoft .NET\Framework\v<%version%> ディレクトリを含むように設定する必要があります。また、<.*NETFrameworkSDKRoot*>*Bin* ディレクトリにある CorVars.bat の実行が必要になることもあります。

参照

その他の技術情報

Visual J# アプリケーションのサンプル

WordCount サンプル (.NET Framework クラス ライブラリの概要)

Download sample

このサンプルは、.NET Framework クラス ライブラリの概要を示します。このサンプルでは、コマンド ラインで指定した複数のファイルを開き、各ファイルのサイズ (バイト数)、文字数、単語数、および行数をカウントするアプリケーションの作成方法を示します。ファイルごとの結果と、すべてのファイルの合計が表示されます。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

開発環境でサンプルをビルドして実行するには

1. [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。
2. F5 キーを押して、サンプルを実行します。

コマンド ラインでサンプルをビルドして実行するには

1. 「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

2. 「WordCount」と入力し、**Enter** キーを押します。
3. ファイル内の単語数をカウントするには、「**WordCount** *filename*」と入力します。

サンプルのクラスとテクノロジ

このサンプルでは、次のクラスとテクノロジが使用されています。

入出力

クラス/テクノロジ	説明
FileStream	ファイルのアクセスに使用されます。このクラスは、ファイルの読み取りと書き込みのために使います。

StreamWriter	出力をファイルに送るようにコマンド ライン引数で指定された場合に使います。**StreamWriter** は、ファイルへのテキスト出力の書式を設定するために、**FileStream** インスタンスと組み合わせて使います。
StreamReader	単語数をカウントするファイルの内容を読み取るために使います。**StreamReader** は、ファイルからテキストを読み取るために、**FileStream** インスタンスと組み合わせて使います。

コレクション

クラス/テクノロジ	説明
ArrayList	オブジェクトのセットを格納するための汎用のコレクション クラスとして使います。
SortedList	オブジェクトのセットを並べ替えて格納するためのコレクション クラスとして使います。
IEnumerator	オブジェクトのセットを列挙するために直接使います。
IDictionaryEnumerator	インデックス付きのオブジェクトのセットを列挙するために直接使います。

参照

その他の技術情報

Visual J# アプリケーションのサンプル

Visual J# の複数の言語を含むサンプル

このセクションのサンプルは、他の言語のサンプルから移植されたサンプルです。

このセクションの内容

サンプル	説明
Delegates サンプル (カスタム デリゲート)	Visual C++ でのデリゲートの使い方を示します。Visual J# で作成された DLL に定義されているメソッドに対し、デリゲートが作成されます。
ManagedEvents サンプル (マネージ アプリケーションでのイベントの作成および使用の例)	C++ マネージ拡張と Visual J# オブジェクトとの間のイベント処理の相互運用性を示します。Visual J# コードは、イベント ソースとイベント レシーバの両方を実装します。
TilePuzzle サンプル : Visual J# と C++ マネージ拡張との相互運用	C++ マネージ拡張および Visual J# で記述されたマネージ コンポーネントと、COM 属性を使って C++ で記述されたネイティブ コンポーネントとの間の相互運用性を示します。

関連するセクション

Visual J# のサンプル

Visual J# アプリケーションのサンプル

Visual J# の技術サンプル

Delegates サンプル (カスタム デリゲート)

Download sample

このサンプルは、**__delegate** キーワードを使って、2 種類のデリゲート (シングルキャストとマルチキャスト) を実装します。

サンプルでは、Visual C++ アプリケーションと Visual J# DLL (`jsdel`) との間の相互運用性も示しています。サンプルの Visual J# DLL は、カスタム メソッドを使って基本的な Visual J# クラス (`jsdel`) を実装します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

Visual Studio でサンプルをビルドして実行するには

1. ソリューション エクスプローラで、**Delegates** ソリューションを右クリックします。
2. ショートカット メニューの [ソリューションのビルド] をクリックします。
3. [デバッグ] メニューの [開始] をクリックします。

サンプルをビルドして実行すると、シングルキャストとマルチキャストのデリゲートが作成され、呼び出されます。

- `pSCDelegate`

  シングルキャスト デリゲートが作成された後、バインドされたメソッドが呼び出されます。呼び出しを実行すると、シングルキャスト デリゲートから渡された値が文字列として出力されます。

- `pMCDelegate`

  マルチキャスト デリゲートを作成する前に、まず Visual J# クラス (`jsdel`) がインスタンス化されます。デリゲートは、Visual J# とマネージ クラスの両方のメソッドにバインドされ、呼び出されます。呼び出しを実行すると、マルチキャスト デリゲートから渡された値が文字列として出力されます。

参照

その他の技術情報

Visual J# の複数の言語を含むサンプル

ManagedEvents サンプル (マネージ アプリケーションでのイベントの作成および使用の例)

Download sample

このサンプルでは、Visual C++ と Visual J# オブジェクトとの間のイベント処理の相互運用性を示します。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

ソリューションをビルドおよび実行するには

1. ソリューション エクスプローラで、[Receiver] ノードを右クリックし、[スタートアップ プロジェクトに設定] をクリックします。
2. [ビルド] メニューの [ソリューションのビルド] をクリックします。
3. [デバッグ] メニューの [デバッグなしで開始] をクリックします。
4. JSEvent をスタートアップ プロジェクトとしてサンプルを実行することもできます。JSEvent をスタートアップ プロジェクトに設定します。
5. [JSEvent] プロジェクトを右クリックし、[プロパティ] をクリックします。[全般] タブで、出力の種類を [コンソール アプリケーション] にします。
6. サンプルをビルドして実行します。

サンプルの動作

コードによって、2 つのイベント ソースが作成されます。Visual C++ のイベント ソース `Event` (Event.cpp を参照) と、Visual J# のイベント ソース `JSEvent` (JSEvent.jsl を参照) です。また、2 つのイベント レシーバも作成されます。Visual C++ のイベント レシーバ `Receiver` (Receiver.cpp を参照) と Visual J# のイベント レシーバ `JR2` (JSEvent.jsl を参照) です。

Visual C++ のイベント ソース `Event` は、2 つのイベント `MyEvent` と `MyEvent2` を宣言します。Visual J# のイベント ソース `JEvent` は、1 つのイベント `JMyEvent` を宣言します。

Visual C++ のイベント レシーバ `Receiver` は、次のイベント メソッドをフックします。

- `Event::MyEvent` をイベント ハンドラ メソッド `Handler1` と `Handler2` にフックします。
- `Event::MyEvent2` をイベント ハンドラ メソッド `Handler5` にフックします。
- `JEvent::JMyEvent` をイベント ハンドラ メソッド `Handler3` と `Handler4` にフックします。

Visual J# のイベント レシーバ JR2 は、次の処理を行います。

- `JEvent::JMyEvent` をイベント ハンドラ メソッド `H1` と `H2` にフックします。
- `Event::MyEvent` をイベント ハンドラ メソッド `H3` と `H4` にフックします。

Receiver.cpp のメイン コードでは、イベント ソースと Visual C++ のイベント レシーバをインスタンス化します。その後、イベント ハンドラをイベント ソースにフックし、これらのイベントを発生させるメソッドを呼び出します。

JSEvent.jsl のメイン コードでは、イベント ソースと Visual J# のイベント レシーバをインスタンス化します。その後、イベント ハンドラをフックし、イベントを発生させ、イベント ハンドラをアンフックするメソッドを呼び出します。

Visual J# のイベント レシーバ `JR2` および Visual C++ のイベント レシーバ `Receiver` は、Visual C++ と Visual J# のイベント ソースの両方から、それぞれ `Event` と `JEvent` というイベントを受け取ります。

参照

その他の技術情報

Visual J# の複数の言語を含むサンプル

TilePuzzle サンプル : Visual J# と C++ マネージ拡張との相互運用

Download sample

このサンプルでは、Visual C++ および Visual J# で記述されたマネージ コンポーネントと、COM 属性を使って C++ で記述されたネイティブ コンポーネントとの間の相互運用性を示します。

サンプルは、"Tile Puzzle" という基本的なパズル ゲームを実装します。サンプルは、ビットマップを読み込み、ユーザーによって指定された任意の数のタイルに分け、ランダムな位置に配置します。ユーザーは、タイルを動かすことによってパズルを解いて元の絵に戻します。さらに、サンプルでは、Visual C++ および .NET Framework クラスで書かれた帰納的な検索アルゴリズムを使用してパズルを解くこともできます。

セキュリティに関するメモ :
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

Visual Studio で TilePuzzle をビルドして実行するには

1. ソリューション エクスプローラで、[Puzzle] ソリューションを右クリックします。
2. ショートカット メニューの [ソリューションのビルド] をクリックします。
3. [デバッグ] メニューの [開始] をクリックします。Puzzle プロジェクトがスタートアップ プロジェクトとして設定されていることを確認してください。

解説

プロジェクトが正しくビルドされたら、パズルを解いてみてください。

ゲームの保存および読み込みの機能は実装されていません。

このサンプルは、マネージ .NET Framework オブジェクトからネイティブ COM コンポーネントにアクセスするために、TLBIMP.EXE を使って .NET Framework プロキシ DLL を作成します。

サンプルのクラス

このサンプルでは、次のクラスが使用されています。

クラス	説明
System.Windows.Forms.Form	Puzzle プロジェクトで使用される `AboutForm` オブジェクトを実装します。
System.Object	Puzzle プロジェクトで使用される `GameLevelEnum` オブジェクトを実装します。

System.Delegate	Puzzle プロジェクトで使用される `SolveThreadProcDlg` オブジェクトを実装します。

参照

その他の技術情報

Visual J# の複数の言語を含むサンプル

Visual J# の技術サンプル

このセクションのサンプルでは、.NET Framework および Visual J# のテクノロジと機能の一部について説明します。各サンプルは、1 つのテクノロジまたは関連するいくつかのテクノロジについて説明することを目的としています。

サンプル	説明
ClassLoader サンプル (アセンブリのカスタム クラス ローダーの変更)	クラス ローダー
CLSExtender サンプル (.NET Framework のプロパティ、イベント、およびデリゲートの作成)	CLS エクステンダ
CrossLanguage サンプル (Visual J# とその他のマネージ言語との相互運用)	複数言語
列挙型のサンプル (列挙型の宣言)	列挙型
bean スタイルのインデックス付きプロパティのサンプル (プロパティ値の設定)	bean スタイルのインデックス付きのプロパティ
Properties サンプル (.NET Framework プロパティの使用)	.NET Framework の使用
bean スタイルの簡単なプロパティのサンプル (プロパティ値の設定)	bean スタイルの簡単なプロパティ
値型のサンプル (ポイント型を値型として宣言)	値型
WFCUsingWinForm サンプル (Visual J++ 6.0 で作成した WFC アプリケーションの Windows フォームによる拡張)	Windows フォームと WFC アプリケーションの拡張

関連するセクション

Visual J# のサンプル

Visual J# アプリケーションのサンプル

Visual J# の複数の言語を含むサンプル

Visual J# の高度な技術サンプル

bean スタイルのインデックス付きプロパティのサンプル (プロパティ値の設定)

Download sample

このサンプルでは、bean スタイルのプロパティを作成する方法を示します。

ボタン型は次の 2 つのプロパティを宣言します。

- **Point** 型。インデックス付きプロパティを宣言します。
- **Point** 型。インデックスを付けて中に関連フィールドを設定できます。

セキュリティに関するメモ :
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

1. [ビルド] メニューの [ソリューションのビルド] をクリックします。

   または

   コマンド ラインで、「**BUILD.bat**」と入力します。

メモ :
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

2. コマンド ラインで、「**app**」と入力します。
3. 出力を調べます。次のような結果が出力されます。

```
p.x: 5
p.y: 10
```

参照

その他の技術情報

bean スタイルの簡単なプロパティのサンプル (プロパティ値の設定)
Visual J# の技術サンプル

bean スタイルの簡単なプロパティのサンプル (プロパティ値の設定)

Download sample

このサンプルでは、bean スタイルのプロパティを作成する方法を示します。ボタン型は次の 2 つのプロパティを宣言します。

- **text** プロパティは単純なプロパティです。
- **pressed** プロパティは、**boolean** プロパティです。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

1. [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

   または

   コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

2. [デバッグ] メニューの [デバッグ開始] をクリックするか、または F5 キーを押します。

   または

   コマンド ラインで、「**app**」と入力します。

3. 出力を調べます。次のような結果が出力されます。

```
Is the button pressed: True
Button text is: Ok
```

参照

その他の技術情報

bean スタイルのインデックス付きプロパティのサンプル (プロパティ値の設定)

Visual J# の技術サンプル

ClassLoader サンプル (アセンブリのカスタム クラス ローダーの変更)

Download sample

このサンプルでは、既存の Java 言語アプリケーションのカスタム クラス ローダーを変更して、マネージ アセンブリからクラスを読み込む方法を示します。新しいコードでマネージ アセンブリからクラスを検索して読み込むには、カスタム クラス ローダーではなく、共通言語ランタイムのセマンティクスと .NET Framework API を使うことが推奨されています。クラス ローダーが使用される状況として想定されているのは、既存の Java 言語アプリケーションによって使用される場合だけです。**java.lang.ClassLoader** クラスの **resolveClass** メソッドと **defineClass** メソッドはサポートされていません。既存の Java 言語アプリケーションでは、**Class.forName** のいずれかのバージョンを呼び出すように、**loadClass** メソッドを書き直す必要があります。その後、アプリケーション構成ファイル (.config) に適切なエントリを登録します。これにより、アプリケーションに必要なマネージ アセンブリを検索して読み込むことができるようになります。マネージ アセンブリがアプリケーションの作業ディレクトリに存在する場合、構成ファイルは不要で、クラスの検索は **Class.forName** の検索ヒューリスティックによって行われます。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

- 開発環境では、F5 キーを押して、サンプルをビルドおよび実行します。

  または

  コマンド ラインで、「**ClassLoaderSample.exe**」と入力します。

参照

その他の技術情報

Visual J# の技術サンプル

CrossLanguage サンプル (Visual J# とその他のマネージ言語との相互運用)

Download sample

このサンプルでは、Visual J# で他の言語で記述されたクラスに Java 言語構文を使ってアクセスする方法と、他の言語から Visual J# クラスにアクセスする方法を示します。また、Java 言語で記述されたコンポーネントが Java 言語例外をスローしたときに、その例外を他の言語でキャッチする方法も示します。このサンプルでは、3 つのアセンブリが作成されます。これらのライブラリ アセンブリまたは DLL アセンブリは、Visual C++、Visual Basic、および Visual J# で記述された単純なクラスを定義します。各クラスは他のクラスから派生しており、根本的には Visual C++ で記述された基本クラスが基になっています。最終的に、C# で記述された実行可能ファイルによって、Visual J# クラスのインスタンスが作成され、そのメソッドが呼び出されます。.NET Framework は、複数の開発者がさまざまな開発言語を使って共同作業をシームレスに行うことができる環境です。

セキュリティに関するメモ：

このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックします。

  または

- コマンド ラインで「**BUILD.bat**」と入力します。

  メモ：

  詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

メモ：

サンプルのコンパイルに必要な DLL の 1 つは、ビルド スクリプトで C:\Windows\Microsoft .NET\Framework\v<バージョン番号>\vjslib.dll としてハードコーディングされています。これは、コンピュータにインストールされている Visual J# のバージョンと Windows のディレクトリ名に基づいて変更することが必要な場合もあります。

このサンプルを実行するには

- 開発環境で F5 キーを押します。

または

コマンド ラインで「**CrossLang.exe**」と入力します。

インスタンス化された各オブジェクトに対して、最低 2 種類の開発言語で記述されたコードが呼び出されます。

参照

その他の技術情報

Visual J# の技術サンプル

列挙型のサンプル (列挙型の宣言)

Download sample

ここでは、列挙型 **color** の宣言例を示します。列挙定数とは明示的に割り当てられる値です。サンプルでは、**enum** 型での各種操作について説明します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

1. [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

   または

   コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

2. コマンド ラインで、「**app**」と入力します。
3. 出力を調べます。次のような結果が出力されます。

```
The Color type is derived from System.Enum
Underlying type: System.Int32
Red
Does c1 equal c2: true
Does c1 equal c2: true
The Color is Red
```

4.

参照

その他の技術情報

Visual J# の複数の言語を含むサンプル
Visual J# の技術サンプル

値型のサンプル (ポイント型を値型として宣言)

Download sample

このサンプルでは、"ポイント" 型が値型として宣言されます。次に値型がインスタンス化され、メソッドが呼び出され、パラメータとしてメソッドに渡されます。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

1. ソリューション エクスプローラの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

   または

   コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

2. [デバッグ] メニューの [開始] をクリックします。

   または

   コマンド ラインで、「**app**」と入力します。

3. 出力を調べます。次のような結果が出力されます。

   ```
   The Point type is derived from System.ValueType
   Does p1 equal p2: true
   Does p1 equal p2: false
   ```

4.

参照

その他の技術情報

Visual J# の複数の言語を含むサンプル

Visual J# の技術サンプル

WFCUsingWinForm サンプル (Visual J++ 6.0 で作成した WFC アプリケーションの Windows フォームによる拡張)

Download sample

このサンプルでは、Windows フォームを使って、Visual J++ 6.0 で作成した WFC アプリケーションを拡張する方法を示します。Visual J++ 6.0 で作成した単純な WFC アプリケーションを Visual J# にアップグレードします。このサンプルは、ボタンまたはメニュー オプションのクリック時に Windows フォーム コンポーネントを起動するように拡張されます。同様の方法を使うと、WFC アプリケーションから任意の Windows フォーム コンポーネントを呼び出して起動できます。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- 開発環境では、F5 キーを押します。

  または

  コマンド ラインで、「**WFCUsingWinForm.exe**」と入力します。

WFC アプリケーションを使うには

- ボタンをクリックして、Windows フォーム コンポーネントを起動します。

  または

[File] メニューの [Start Windows Form] をクリックして、同じ Windows フォーム コンポーネントを起動します。

参照

その他の技術情報

Visual J# の技術サンプル

Visual J# アプリケーションの高度なサンプル

このセクションのサンプルは、完成済みの一般的なアプリケーションにならって作成されています。「Visual J# アプリケーションのサンプル」で紹介されているサンプルよりも難易度の高いアプリケーション サンプルが紹介されています。

このセクションの内容

Acronym-WebService

XML Web サービスを作成し、Web アプリケーションから利用する例を示します。データセットの内容をサーバー側のデータ グリッドに連結する方法も示します。

AgentExplorer

Java 言語/COM アプリケーションをアップグレードして Visual J# で実行する方法を示します。

Vjsresgen

Visual J++ 6.0 のリソース ファイルを .NET Framework のリソース ファイル (.resx) に変換します。

関連するセクション

Visual J# のサンプル

Visual J# アプリケーションのサンプル

Visual J# の技術サンプル

Visual J# の複数の言語を含むサンプル

Visual J# の複数の言語を含む高度なサンプル

Visual J# の高度な技術サンプル

Acronym-WebService サンプル (XML Web サービスの作成と利用)

Download sample

このサンプルでは、次の方法を示します。

- XML Web サービスの作成
- ボタン、ラベル、データ グリッドなどの Web コントロールの使用
- Web サービスの利用

この Web サービスは、頭字語の最初の何文字かを受け取り、一致する頭字語をすべて返します。

Web アプリケーションは、この Web サービスを利用して、データセットの内容をサーバー側の Datagrid コントロールにバインドします。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

必要条件

- IIS
- Microsoft Visual Studio
- Microsoft Visual J#

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

サンプルを実行できるようにするには

1. サンプル ディレクトリ (AcronymWA および AcronymWebServiceInJavaLanguage) をコンピュータにダウンロードしたら、%SystemDrive%\inetpub\wwwroot にコピーします。
2. 次の操作を行って、AcronymWA を IIS の仮想ディレクトリにします。
   a. [スタート] ボタンをクリックし、[プログラム] をポイントします。次に、[管理ツール] をポイントし、[インターネット サービス マネージャ] をクリックします。
   b. ツリーの [既定の Web サイト] を右クリックし、[新規作成] をポイントして、[仮想ディレクトリ] をクリックします。
   c. エイリアスを "AcronymWA" に設定します。
   d. ディレクトリを %SystemDrive%\inetpub\wwwroot\AcronymWA に設定します。

e. アクセス許可のオプション ([読み取り]、[ASP 等のスクリプトを実行する]、[ISAPI アプリケーションや CGI 等を実行する]、[書き込み]、および [参照] の各チェック ボックス) をすべてオンにします。

3. 同じ手順に従って、AcronymWebServiceInJavaLanguage の仮想ディレクトリを作成します。
4. ローカル ユーザー アカウント ASPNET (<your_machine_name>\ASPNET) の AcronymWebServiceInJavaLanguage サブディレクトリに書き込みアクセス許可を追加します。これにより、このディレクトリにある Microsoft Access データベース Acro.mdb に対して、Web サービスによる読み取りと書き込みができるようになります。

メモ：
WhatItMeans.asmx.jsl ファイルで set_ConnectionString メソッドを検索し、必要に応じて C: を適切なドライブ名に置き換えてください。

5. Microsoft Visual Studio .NET を起動します。
6. %SystemDrive%\inetpub\wwwroot\AcronymWA から AcronymWA.sln を開きます。
7. [ビルド] メニューの [ソリューションのビルド] をクリックします。
8. ソリューション エクスプローラで AcronymExplainer.aspx を右クリックし、[スタート ページに設定] をクリックします。

サンプルをテストするには

1. [デバッグ] メニューの [デバッグなしで開始] をクリックします。頭字語の入力を要求するページが表示されます。Acro.mdb データベースには、Microsoft Access から "vj#sample" データベース パスワードでアクセスできます。
2. エディット ボックスに任意の文字を入力します。
3. [Get Acronym !] をクリックします。
4. タイトルの一覧がデータ グリッドに表示されます。

参照

その他の技術情報

Visual J# アプリケーションのサンプル

AgentExplorer サンプル (Java 言語/COM アプリケーションのアップグレード)

Download sample

このサンプルでは、Visual J++® 6.0 の Java 言語/COM アプリケーションを Visual J# で実行できるようにアップグレードする方法を示します。このサンプルは、JActiveX® ツールによって生成される Microsoft Agent コントロール COM コンポーネントのラッパーと、com.ms.* および JDK レベル 1.1.4 パッケージのクラスを使います。このサンプルは、事前に生成した JActiveX ラッパーに new 演算子を使うことにより、Microsoft® Agent COM コントロールのインスタンスを作成します。次に、COM オブジェクトを特定のインターフェイスにキャストして、メソッドやプロパティを呼び出します。JActiveX ラッパーは、%WinDir%\msagent ディレクトリにある COM サーバーの agentsvr.exe から生成されます。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

開発環境でビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

コマンド ラインでビルドするには

- 「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- 開発環境では、F5 キーを押します。

  または

  コマンド ラインでは、「**AgentExplorer**」と入力し、Enter キーを押します。

参照

その他の技術情報

Visual J# アプリケーションのサンプル

Vjsresgen サンプル (Visual J++ 6.0 リソース ファイルの .NET Framework リソース ファイル (.resx) への変換)

Download sample

このサンプルでは、Visual J++ 6.0 のリソース ファイルを .NET Framework のリソース ファイル (.resx) に変換する方法を示します。

アップグレードされたアプリケーションが正しく動作するようにするには、アプリケーションで使用されるすべてのリソースが CLASSPATH から識別され、アセンブリに埋め込まれるかリンクされることが重要です。リソース ファイルを .NET Framework リソースに変換するときは、ディレクトリ構造が保持されるように注意する必要があります。

セキュリティに関するメモ：

このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

  メモ：

  詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- このサンプルは、コマンド ラインから実行する必要があります。コマンド ラインで「**vjsresgen.exe**」と入力し、.resx 形式に変換するファイル名を指定します。

  既定では、生成される .resx ファイルの名前は vjs.resx になります。**/out** オプションを使うと、この名前をオーバーライドできます。

Vjsresgen の構文とオプション

```
vjsresgen [/out:filename] [/recurse:wildcard] resourcefiles
```

解説

指定項目 :

filename

作成する .resx 出力ファイルの名前。

wildcard

ファイル名に一致するワイルドカード表現。

resourcefiles

入力リソース ファイル。

/recurse オプションを使うと、ワイルドカード表現 (*wildcard*) に一致するファイルを指定できます。オプションとして、検索を開始するディレクトリを指定することもできます。指定しない場合は、プロジェクト ディレクトリから検索されます。

Vjsregen.exe の実行例

コマンド ラインで Vjsresgen.exe を使う方法を次の例に示します。

同じディレクトリ内のリソース

- Vjsresgen /out:myresource.resx *.txt
- Resgen myresource.resx myresource.resources
- JbImp /res:myresource.resources myarchive.zip

  または

  vjc /resource:myresource.resources *.java

サブディレクトリ内のリソース

- Vjsresgen /out:myresource.resx subdir\res1.bmp subdir\r.resources subdir2\res.txt subdir2\r.resources
- Resgen myresource.resx myresource.resources
- JbImp /res:myresource.resources myarchive.zip

  または

  vjc /resource:myresource.resources *.java

  上の例の subdir\r.resources と subdir2\r.resources は、2 つの異なるリソース ファイルです。パッケージ subdir に存在するクラスの **Class.getResource** を呼び出すと、subdir\r.resources で指定されるリソースが取得されます。パッケージ subdir2 のクラスでは、subdir2\r.resources が取得されます。

  上の例の r.resources は Visual J++ 6.0 で作成されたファイルで、.NET Framework でサポートされている .resources ファイル形式とは異なります。

/recurse オプションを使ったリソース

- Vjsresgen /out:my.resx /recurse:p1*.resources /recurse:p2*.bmp
- Resgen my.resx

  さらに、次のようになります。

- JbImp /res:my.resources /recurse

  または

  vjc /resource:my.resources /recurse:*.java

  上の例では、ディレクトリ p1 に存在する *.resources とディレクトリ p2 に存在する *.bmp ファイルが埋め込まれます。

メモ :

マネージ リソース ファイル (.resx) に埋め込まれた変換済みのリソース ファイルへのパスは、**Class.getResource** (または類似のメソッド) でリソースを指定するために使用されるパスと一致している必要があります。このため、アプリケーション コードを適切に変更することが必要になる場合があります。たとえば、C:\resources\package1\data.txt にある Visual J++ リソースを変換するには、C:\temp ディレクトリに移動してから、Vjsresgen を実行します。次に、`Class.getResources(String` *resourceName*`)` などのメソッドで "package1.data.txt" としてリソースを指定するように、アプリケーション コードを変更する必要があります。

参照

処理手順

方法 : リソースを使う Visual J++ 6.0 アプリケーションのアップグレード

その他の技術情報

Visual J# アプリケーションのサンプル

Visual J# の複数の言語を含む高度なサンプル

次のサンプルは、他の言語のサンプルから移植されたサンプルです。

このセクションの内容

DataSetConsumer サンプル (複数テーブルのデータセットの利用)

SPROXY で生成された SOAP クライアントで複数テーブルのデータセットを使う方法を示します。Visual J# コードは、複数テーブルのデータセットを作成し、XML Web サービス メソッドを通じて公開します。

JrnlPost サンプル (COM クライアントおよび .NET Framework クライアントからのビジネス ロジックへのアクセス)

ネイティブ ビジネス ロジック、COM クライアント用のインターフェイスとタイプ ライブラリ、および対応するメタデータを持つ .NET Framework クライアント用のアセンブリを含む単一の DLL を作成します。Visual J# コードは、DLL からマネージ オブジェクトをインスタンス化し、そのオブジェクトに対していくつかのメソッドを呼び出します。DLL に COM クラスと .NET Framework クラスの両方が含まれていることを示します。

MCppWrapper サンプル (C++ DLL のラップ)

マネージ拡張 __gc クラスを DLL 内のアンマネージ C++ クラスのプロキシ クラス (ラッパー クラス) として使用する方法を示します。Visual J# コードは、コンソールから文字列と位置を読み取り、サフィックスを表示するクライアントです。

関連するセクション

Visual J# のサンプル

Visual J# アプリケーションのサンプル

Visual J# の技術サンプル

Visual J# の複数の言語を含むサンプル

Visual J# の高度な技術サンプル

Visual J# アプリケーションの高度なサンプル

DataSetConsumer サンプル (複数テーブルのデータセットの利用)

Download sample

このサンプルは、XML Web サービス プロキシを使って、アンマネージ Visual C++ クライアントから複数テーブルのデータセットを利用する方法を示します。

DataSetConsumer サンプルでは、アンマネージ Visual C++ アプリケーションで複数テーブルのデータセットを使う方法を示します。DataSetConsumer は、複数テーブルのデータセットを使う ATL Server アプリケーションです。データセットは、ASP.NET を使って作成された XML Web サービスを通じて公開されます。

また、DataSetConsumer は、複数テーブルのデータセットの使用をサポートするための Web 参照の使い方も示します。Web 参照では、単一テーブルのデータセットは C++ 構造体に変換されますが、複数テーブルのデータセットは XML ストリームとして扱われます。このため、プロキシ クラス内の対応する XML Web サービス メソッドも、データセットの完全な XML シリアル化を文字列データとして返します。ほとんどのアプリケーションでは、このデータを使うために解析が必要になります。

DataSetConsumer アプリケーションでは、1 つのデータセット内の複数のテーブルにアクセスします。一方、チュートリアルでは、単一のテーブルにアクセスします。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

必要条件

- IIS をインストールして実行する必要があります。
- 少なくとも 1 つの SQL Server データベースをインストールする必要があります。

サンプルのビルドと実行

このサンプルを使用するには、ここで説明する手順を実行してください。ここでは、XML Web サービス プロジェクト WebService1 を作成し、データセットのデータにアクセスする Web サービス メソッドを定義します。次に、XML Web サービスのプロキシ クラスを作成する Web 参照を追加します。DataSetConsumer は、プロキシのインスタンスを使って `WebService1` の XML Web サービス メソッドを呼び出します。`WebService1` は、データセットを **BSTR** 型として返します。DataSetConsumer は、これを XML として読み込み、DOM および MSXML パーサーを使って文字列を解析し、データを HTML として出力します。

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

データセットにアクセスする ASP.NET Web サービス プロジェクトを作成するには

1. ASP.NET を使用して XML Web サービスを新規作成します。

   [ファイル] メニューの [新規作成] をポイントし、[プロジェクト] をクリックします。[新しいプロジェクト] ダイアログ ボックスが表示されます。[Visual J# プロジェクト] フォルダの [ASP.NET Web サービス] をクリックします。プロジェクトに `WebService1` という名前を付けます。位置は既定値 (`http://localhost/WebService1`) にします。

[OK] をクリックします。プロジェクトを作成すると、[デザイン] ペインに Service1.asmx.jsl が表示されます。

2. データ接続を作成します。

   サーバー エクスプローラで、[データ接続] を右クリックし、[接続の追加] をクリックします。[データ リンク プロパティ] ダイアログ ボックスが表示されます。このダイアログ ボックスで設定を行い、データベース (たとえば Northwind データベース) へのデータ接続を作成します。[パスワードを保存する] チェック ボックスをオンにします。

3. テーブルおよびデータアダプタを追加します。

   新しいデータ接続の [テーブル] ノードを開き、サーバー エクスプローラから [デザイン] ペインに 2 つのテーブルをドラッグします。2 つの DataAdapter と 1 つのデータ接続が [デザイン] ペインに表示されます。

4. データセットを生成します。

   [データ] メニューの [データセットの生成] をクリックします。[データセットの生成] ダイアログ ボックスで、[新規作成] (既定の名前は `DataSet1`) をクリックし、両方のテーブルをデータセットに追加します。また、[このデータセットをデザイナに追加する] ボックスをオンにすることをお勧めします (省略可能)。[OK] をクリックして、データセットを作成します。既定の名前は `dataSet11` です。

5. Web サービス メソッドを実装します。

   [表示] メニューの [コード] をクリックします。Service1.asmx.jsl のコードが [編集] ペインに表示されます。ファイルの終わりまでスクロールして、`HelloWorld` というサンプルの Web サービス メソッドを見つけます。

```
// /** @attribute WebMethod() */
//    public System.String HelloWorld()
//    {
//        return "Hello World";
//    }
```

   `HelloWorld` サンプル コードの代わりに、次のように `GetDataSet` という新しい Web サービス メソッドを実装します。

```
    /** @attribute WebMethod() */
    public DataSet1 GetDataSet()
    {
        this.sqlDataAdapter1.Fill (this.dataSet11);
        return dataSet11;
    }
```

メモ :
このコードでは、SQL Server データベースの使用を想定しています。

6. WebService1 プロジェクトをビルドします。[ビルド] メニューの [ソリューションのビルド] をクリックします。

DataSetConsumer のデータを使用するには

1. DataSetConsumer プロジェクトを開きます。
2. Visual Studio 開発環境で、ソリューション ファイル DataSetConsumer.sln を開きます。
3. DataSetConsumer.h に次の行が含まれていることを確認します。

```
#include "WebService.h"
```

4. プロジェクトに Web 参照を追加します。

   Web 参照を追加すると、XML Web サービスのプロキシ クラスが生成されます。XML Web サービス クライアントは、プロキシのメソッドを呼び出すことができます。このメソッドは、ネットワーク経由でリモート呼び出しを実行するために必要な作業を行い、対応する XML Web サービス メソッドを呼び出します。既定では、プロキシ クラスは SOAP を使って対応するメソッドにアクセスします。サポートされている 3 つのプロトコル (SOAP、HTTP-GET、および HTTP-POST) のうち、SOAP はサポートするデータ型が最も豊富です。Web 参照を追加すると、ほかの 2 つのプロトコルにも同様にプロキシ クラスが生成されます。ソリューション エクスプローラで、[DataSetConsumer] プロジェクト

ノードを右クリックし、ショートカット メニューの [Web 参照の追加] をクリックします。

[Web 参照の追加] ダイアログ ボックスが表示されます。

5. [アドレス] バーに、次に示す WebService1 の URL を入力し、**Enter** キーを押します。

```
http://localhost/WebService1/Service1.asmx
```

Web サービス記述は、Web サービス記述言語 (WSDL: Web Service Description Language) と呼ばれる XML 文法で書かれた XML ドキュメントです。XML Web サービスのメッセージの書式は、WSDL で定義されます。

6. [参照の追加] をクリックします。これにより、プロキシ クラス `Service1` が DataSetConsumer プロジェクトに追加されます。ソリューション エクスプローラには、ヘッダー ファイル WebService.h が表示されます。

メモ :

Web サービス用に生成されたヘッダー ファイルでは、`CService1T` コンストラクタは、Web 参照を追加したときに指定した URL を使うようにハードコーディングされています。別の XML Web サービスで DataSetConsumer を使う場合は、XML Web サービスの .asmx ファイルの場所と一致するように URL を変更してください。

7. [ビルド] メニューの [ソリューションのビルド] をクリックして、DataSetConsumer プロジェクトをビルドします。
8. クライアント アプリケーションを実行するには、[デバッグ] メニューの [デバッグなしで開始] をクリックします。

   URL. の入力を求めるプロンプトが表示されます。
9. 次の URL を入力します。

```
http://localhost/DataSetConsumer/DataSetConsumer.srf
```

10. [OK] をクリックします。

    Web ブラウザが表示されて、XML Web サービスを通じてアクセスされたデータが表示されます。

サンプルのクラス、テクノロジ、および概念

- Web 参照の追加と削除
- ADO.NET DataSet
- XML Web サービス クライアントの作成
- DataSet クラス
- request_handler
- soap_handler
- soap_method

参照

その他の技術情報

Visual J# の複数の言語を含むサンプル

JrnlPost サンプル (COM クライアントおよび .NET Framework クライアントからのビジネス ロジックへのアクセス)

Download sample

このサンプルでは、ネイティブ ビジネス ロジック、COM クライアント用のインターフェイスとタイプ ライブラリ、および対応するメタデータを持つ .NET Framework クライアント用のアセンブリを含む、1 つの DLL を作成します。Visual J# および C++ マネージ拡張で記述されたクライアントを使って、ビジネス ロジックにアクセスします。

セキュリティに関するメモ :

このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックします。

  または

  ソリューション エクスプローラで、[JrnlPost] ソリューションを右クリックし、ショートカット メニューの [ソリューションのビルド] をクリックします。

このサンプルを実行するには

1. ソリューション エクスプローラで、[comJrnlClient] プロジェクトを右クリックします。
2. ショートカット メニューの [デバッグ] をポイントし、[新しいインスタンスを開始] をクリックします。
3. netJrnlClientA プロジェクトと netJrnlClientB プロジェクトで手順 1. と 2. を繰り返します。

クライアントが開始されると、[OK] ボタンのある一連のダイアログ ボックスが表示され、ビジネス ロジックの呼び出しが行われていることが通知されます。[OK] をクリックしてダイアログ ボックスを閉じます。

メモ :

コンポーネント間の対話をより詳しく理解するには、ブレークポイントを設定するか、[デバッグ] メニューの [ステップ イン] または [ステップ オーバー] を使用して、コードを 1 ステップずつ実行します。

プロジェクトの説明

このサンプルでは、ビジネス ロジック自体の実装ではなく、COM クライアントおよび .NET Framework クライアントからのビジネス ロジックへのアク

セスに重点が置かれています。そのため、このサンプルに示されたビジネス ロジックは、比較的簡単なものです。実際に、ビジネス ロジックは最小限の検査だけを行い、サーバーのビジネス アクションに対応した Win32 メッセージ ボックスを表示して手順を示します。ビジネス ロジックを実装する `JEPost` クラスは、JEPost.h で定義され、JEPost.cpp で実装されています。

COM コンポーネントを実装するクラスには、ビジネス ロジック クラスのインスタンスへのポインタがあります。つまり、COM コンポーネントは、基になるビジネス ロジックに COM クライアントがアクセスするための接続として利用されます。COM クライアントは、JrnlPost.dll に含まれているタイプ ライブラリを使用して、COM コンポーネントが公開するインターフェイスやメソッドを探索します。このサンプルの COM クライアントは、コンパイラの **#import** 機能を使ってタイプ ライブラリを利用し、スマート ポインタを生成します。スマート ポインタは、COM コンポーネントのインスタンス化と呼び出しのためにクライアントによって使用されます。COM コンポーネントを実装するクラスは `comJEPost` です。このクラスは comJEPost.h で定義され、comJEPost.cpp で実装されています。

.NET Framework コンポーネントを実装するクラスにも、ビジネス ロジック クラスのインスタンスへのポインタがあります。このサンプルの .NET Framework コンポーネントは、COM の場合と同様に、基になるビジネス ロジックに .NET Framework クライアントがアクセスするための接続として利用されます。JrnlPost.dll には、対応するメタデータを持つ単一のファイル アセンブリが含まれます。このメタデータは、.NET Framework コンポーネントが公開するパブリック クラスやパブリック メソッドの探索とアクセスのために、.NET Framework クライアントによって使用されます。.NET Framework コンポーネントを実装するクラスは `netJEPost` です。このクラスは netJEPost.h で定義され、netJEPost.cpp で実装されています。

このサンプルには、3 つのクライアント プロジェクトがあります。

- ComJrnlClient　ネイティブ C++ で実装された COM クライアント
- NetJrnlClientA　Visual J# で作成された .NET Framework クライアント
- NetJrnlClientB　C++ マネージ拡張を使って作成されたクライアント

参照

その他の技術情報

Visual J# の複数の言語を含むサンプル

MCppWrapper サンプル (C++ DLL のラップ)

Download sample

このサンプルでは、マネージ拡張 **__gc** クラスを、DLL 内のアンマネージ C++ クラスに対するプロキシ クラス (またはラッパー) として使う方法を示します。このサンプルによって示される基本概念には、マネージ拡張の相互運用性、ラッパー クラスとプロキシ クラス、基本的なデータ マーシャリングなどがあります。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

Visual Studio でサンプルをビルドして実行するには

1. ソリューション エクスプローラで、[MCppWrapper] ソリューションを右クリックします。
2. ショートカット メニューの [ソリューションのビルド] をクリックします。
3. [デバッグ] メニューの [開始] をクリックします。

サンプルの動作

アンマネージ DLL には、`substring` というクラスがあります。このクラスにある `suffix` というメソッドは、インデックス位置から文字列の末尾までを文字列のサフィックスとして返します。文字列の最初の文字のインデックス位置は 1 になります。0 または負の位置を指定すると、文字列全体が返されます。文字列の長さを超えた位置を指定すると、空の文字列が返されます。

部分文字列クラスは、`substring_w` というマネージ拡張 **__gc** クラスによってラップされます。このクラスには、マネージ文字列と **int** を引数として受け取る `find_suffix` というメソッドがあります。このメソッドは、簡単なデータ マーシャリングを実行して、マネージ文字列をアンマネージ文字列に変換します。メソッドは `substring` クラスのオブジェクトを作成し、変換された文字列と整数の位置を使用して `suffix` を呼び出します。返される文字列は、暗黙にマネージ文字列に変換されます。

`substring_w` クラスのクライアントは、コンソールから文字列と位置を読み取った後、`substring_w` オブジェクトを作成し、`find_suffix` を呼び出してサフィックスを表示する簡単な Visual J# プログラムです。

参照

その他の技術情報

Visual J# の複数の言語を含むサンプル

Visual J# の高度な技術サンプル

このセクションのサンプルでは、.NET Framework および Visual J# のさまざまなテクノロジと機能を示します。各サンプルは、1 つのテクノロジまたは関連するいくつかのテクノロジについて説明することを目的としています。

このセクションの内容

Attributes サンプル (.NET Framework 属性の使用)

.NET Framework の使用

参照渡しのサンプル (型の参照渡し)

型を参照渡しでメソッドに渡すサンプル

CLSExtender サンプル (.NET Framework のプロパティ、イベント、およびデリゲートの作成)

CLS エクステンダ

CodeDOM サンプル (CodeDomProvider の使用)

CodeDOM

COMCallsJava_NET サンプル (.NET Framework 相互運用機能を使った COM へのマネージ コンポーネントの公開)

Java 言語/COM (.NET Framework セマンティクスの使用)

COMCallsJava サンプル (Visual J++ 6.0 Java 言語/COM コンポーネントのアップグレード)

Java 言語/COM

カスタム属性のサンプル (カスタム属性のクラスへの追加)

カスタム属性

DynamicLoading サンプル (配置済みアセンブリからのクラスの読み込み)

動的読み込み

EventsAndDelegates サンプル (.NET Framework イベントおよびデリゲートの使用)

.NET Framework の使用

ジェネリックのサンプル (ジェネリックの使用)

ジェネリックの使用

JavaCallsCOM サンプル (Visual J++ 6.0 Java 言語/COM コンポーネントのアップグレード)

Java 言語/COM

JavaCallsCOM_NET サンプル (.NET Framework 相互運用機能を使ったマネージ コンポーネントへの COM オブジェクトの公開)

Java 言語/COM (.NET Framework セマンティクスの使用)

Minesweeper サンプル (マインスイーパ ゲームのビルド)

.NET Framework の使用

PInvoke サンプル (アンマネージ API の呼び出し)

プラットフォーム呼び出しサービスの使用

RemotingSample サンプル (リモート処理サービスの作成)

リモート処理サービス

Scoping サンプル (/securescoping の使用)

スコープの指定

SecurityProviders サンプル (サードパーティのセキュリティ プロバイダの指定)

セキュリティ プロバイダ

関連するセクション

Visual J# のサンプル

Visual J# アプリケーションのサンプル

Visual J# の技術サンプル

Visual J# の複数の言語を含むサンプル

Visual J# アプリケーションの高度なサンプル

Attributes サンプル (.NET Framework 属性の使用)

このサンプルでは、Visual J# によってコンパイルおよび実行される言語アプリケーションで、.NET Framework の属性を使って機能を強化する方法を示します。Visual J# プログラムには、.NET Framework で使用できる任意の属性を追加できます。ただし、Visual J# コンパイラでは、Visual J# プログラムでの属性の作成 (定義) はサポートされていません。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- 開発環境では、F5 キーを押します。

  または

  コマンド ラインで、「**AttributeSample.exe**」と入力します。

参照

その他の技術情報

Visual J# の技術サンプル

参照渡しのサンプル (型の参照渡し)

Download sample

このサンプルでは、参照型、値型、またはプリミティブ型を参照渡しでメソッドに渡す方法を示します。

サンプルでは、**point** 型が値型として宣言され、値型がインスタンス化されます。メソッドは値型で呼び出され、参照渡しでメソッドに渡されます。

また、**swap** メソッドの例が示すようにプリミティブ型をパラメータとして参照渡しで渡すこともできます。

セキュリティに関するメモ :
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

1. [ビルド] メニューの [ソリューションのビルド] をクリックします。

   または

   コマンド ラインで、「**BUILD.bat**」と入力します。

メモ :
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

2. コマンド ラインで、「**app**」と入力します。

3. 出力を調べます。次のような結果が出力されます。

```
Does p1 equal p2: true
Does p1 equal p2: false
Swapping two values
Before swap: i = 1 j = 101
After swap: i = 101 j = 1
```

参照

その他の技術情報

Visual J# の技術サンプル

CLSExtender サンプル (.NET Framework のプロパティ、イベント、およびデリゲートの作成)

Download sample

このサンプルでは、.NET Framework のプロパティ、イベント、およびデリゲートの作成方法を示します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

サンプルを実行するには

- 開発環境では、F5 キーを押します。

  または

  コマンド ラインで、「**Container.exe**」と入力します。

参照

その他の技術情報

Visual J# の技術サンプル

CodeDOM サンプル (CodeDomProvider の使用)

Download sample

このサンプルでは、Visual J# CodeDomProvider を使ってコードを動的に生成し、CodeDOM を使ってコンパイルする方法を示します。このサンプルは、次の処理を行います。

- 単純な Hello World プログラムの codeDOM ツリーを生成します。
- それに対するソース コードをファイル bin\GeneratedFile.jsl に生成します。
- 生成されたファイルを Visual J# の CodeDomProvider 実装を使ってコンパイルし、"hello world!" アセンブリを生成します。

CodeDOM の詳細については、「CodeDOM クイック リファレンス」を参照してください。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- 開発環境では、F5 キーを押して、サンプルをビルドおよび実行します。

  ファイル GeneratedFile.jsl が、bin\debug ディレクトリに生成されます。

  または

  コマンド ラインで、「**CodeDOMUsage.exe**」と入力します。生成されたファイル GeneratedFile.jsl は、bin\debug ディレクトリから利用できます。

参照

その他の技術情報

Visual J# の技術サンプル

COMCallsJava_NET サンプル (.NET Framework 相互運用機能を使った COM へのマネージ コンポーネントの公開)

Download sample

このサンプルでは、.NET Framework の COM 相互運用セマンティクスを使って、Visual J# で記述されたマネージ コンポーネントを COM に公開する方法を示します。Visual J# コンポーネントを新しく記述するときは、COM 相互運用の方法として .NET Framework セマンティクスを使うことが推奨されています。このサンプルでは、Visual J# で記述されたマネージ コンポーネントを DLL にコンパイルします。次に、RegAsm ツールを使って DLL を COM コンポーネントとして登録します。登録した DLL は、Visual Basic 6.0 などのクライアントからアクセスできるようになります。

.NET Framework セマンティクスを使った COM 相互運用の詳細については、「アンマネージ コードとの相互運用」を参照してください。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

1. Visual Basic 6.0 で VB6Project.vbp ファイルを開きます。
2. [プロジェクト] メニューの [参照設定] をクリックします。次に、サンプル ディレクトリにある MyForm.tlb ファイルへの参照を追加します。このファイルは、IDE からビルドした場合はプロジェクト ルートの \bin\debug サブディレクトリにあります。
3. F5 キーを押します。
4. 実行されたら、Visual Basic フォームのボタンをクリックして、マネージ フォームを作成します。Visual Basic フォームでプロパティを調整し、マネージ フォームに適用される変更を確認します。Visual Basic フォームでは、マネージ フォームにボタンを作成し、そのボタンのプロパティ

を調整することもできます。これらの処理は、すべてアンマネージ Visual Basic コードから実行されます。

参照

その他の技術情報

Visual J# の技術サンプル

COMCallsJava サンプル (Visual J++ 6.0 Java 言語/COM コンポーネントのアップグレード)

Download sample

このサンプルでは、COM DLL として公開される Visual J++ 6.0 の Java 言語コンポーネントをアップグレードする方法について説明します。Visual J++ 6.0 コンポーネントをアップグレードするには、.NET Framework ツールの TlbImp を使って、COM サーバーのタイプ ライブラリからマネージ ラッパー DLL を生成します。ソースのコンパイル時には、このマネージ DLL を参照します。その後、.NET Framework ツールの RegAsm を使って、出力された DLL を登録します。

Visual J++ 6.0 の Java 言語/COM アプリケーションから Visual J# へのアップグレードの詳細については、「COM に公開される Visual J++ 6.0 コンポーネントのアップグレード」を参照してください。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

1. Visual Basic® 6.0 で、COMClient ディレクトリに移動し、COMClient.vbp ファイルを開きます。

   または

   [プロジェクト] メニューの [参照設定] をクリックします。

2. ..\JavaServer\ ディレクトリにある Server.tlb ファイルへの参照を追加します。F5 キーを押して、デモをビルドおよび実行します。

参照

その他の技術情報

Visual J# の技術サンプル

カスタム属性のサンプル (カスタム属性のクラスへの追加)

Download sample

このサンプルでは、カスタム属性である **author** が J# で宣言されます。これによって、この属性をすべてのプログラム要素に追加できます。また、すべてのプログラム要素をターゲットにできます。

次にカスタム属性は **Point** クラスおよび **Point** クラスのコンストラクタに追加されます。

app クラスが **Point** クラスに反映され、カスタム属性が追加されているかどうか、どのプログラム要素に追加されているかが確認されます。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

1. コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

2. コマンド ラインで、「**app**」と入力します。
3. 出力を調べます。次のような結果が出力されます。

   ```
   Number of custom attributes on type Point are: 1
   Number of methods (incl. constructors) on type Point are: 1
   Method name: .ctor
   Number of custom attributes on method: 2
   ```

4.

参照

その他の技術情報

Visual J# の複数の言語を含むサンプル
Visual J# の技術サンプル

DynamicLoading サンプル (配置済みアセンブリからのクラスの読み込み)

Download sample

このサンプルでは、複数の場所に配置されているマネージ DLL から、**Class.forName** API を使ってクラスを読み込む方法を示します。**Class.forName** には次の 2 種類のバージョンがあります。

- **Class.forName(String className)**
- **Class.forName(String assemblyName, String className, boolean absolutePath)**

詳細については、「実行時バインディング : Class.forName セマンティクス」を参照してください。

セキュリティに関するメモ :
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

このサンプルをビルドするには

- コマンド ラインで、「**BUILD.bat**」と入力します。

メモ :
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- コマンド ラインで「**DynamicLoading.exe**」と入力します。

参照

その他の技術情報

Visual J# の技術サンプル

EventsAndDelegates サンプル (.NET Framework イベントおよびデリゲートの使用)

Download sample

このサンプルでは、.NET Framework のデリゲートとイベントの使用例を示します。Visual J# では、.NET Framework デリゲートを利用できます。**delegate** キーワードまたは **multicast** キーワードは、System.Delegate 型ではなく com.ms.lang.Delegate 型のデリゲートを作成します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- 開発環境では、F5 キーを押します。

  または

  コマンド ラインで、「**EventsAndDelegates.exe**」と入力します。

コマンド ライン要件

Visual J# サンプルをコマンド ラインでビルドして実行する場合、サンプルが正しく機能するようにするには、vsvars.bat (<%Program Files%>\Microsoft Visual Studio 8\Common7\Tools ディレクトリ) を実行する必要があります。また、Visual Studio コマンド プロンプトを使用することもできます。コマンド プロンプトを使用するには、[スタート] ボタンをクリックし、[プログラム] をポイントします。次に、[Microsoft Visual Studio] をポイントし、[Visual Studio Tools] をクリックします。

参照

その他の技術情報

Visual J# の技術サンプル

ジェネリックのサンプル (ジェネリックの使用)

ここでは、ジェネリックを C# で作成し、J# で使用するサンプルを示します。

サンプル	説明
ジェネリック スタック クラスのサンプル (J# を使用して処理)	C# で記述され、J# で使用されるジェネリック スタック クラス。
制約のサンプル (ジェネリック クラスを使用した型の制約)	ジェネリック スタックの型によって制約が適用されています。型パラメータの "T" は、インターフェイス `IDisplay` を実装する型である必要があります。
デリゲートのサンプル (ジェネリック デリゲート型を適用)	汎用デリゲート型である `Proc` は、C# で宣言されます。J# から汎用デリゲート型をインスタンス化し、適用します。
ジェネリック スタック インターフェイスのサンプル (インターフェイスを利用)	ジェネリック スタック クラスはジェネリック スタック インターフェイスを実装します。J# から、このジェネリック スタック クラスをジェネリック スタック インターフェイスという概念で使用します。
出力のジェネリック メソッドのサンプル (型の配列を出力)	"プリンタ" 型には型の配列をコンソールに出力できるジェネリック メソッドがあります。この型は、J# から使用され、さまざまな構成された型を提供します。

参照

その他の技術情報

Visual J# のサンプル
Visual J# の複数の言語を含むサンプル
Visual J# アプリケーションのサンプル
Visual J# の技術サンプル

ジェネリック スタック クラスのサンプル (J# を使用して処理)

Download sample

このサンプルでは、C# で記述され、J# で処理されるジェネリック スタック クラスを示します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

参照

その他の技術情報

ジェネリックのサンプル (ジェネリックの使用)
Visual J# の複数の言語を含むサンプル
Visual J# の技術サンプル

制約のサンプル (ジェネリック クラスを使用した型の制約)

Download sample

このサンプルでは、ジェネリック スタックの型によって制約が課せられています。型パラメータの "T" は、インターフェイス **IDisplay** を実装する型である必要があります。スタックの型では、新しいメソッド **displayAll** が導入されています。このメソッドは、スタックを使用して格納された型の **IDisplay** インターフェイスにメソッドを呼び出します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

- [デバッグ] メニューの [デバッグ開始] をクリックするか、または F5 キーを押します。

  または

  コマンド ラインで、「**app**」と入力します。

参照

その他の技術情報

ジェネリックのサンプル (ジェネリックの使用)
Visual J# の複数の言語を含むサンプル
Visual J# の技術サンプル

デリゲートのサンプル (ジェネリック デリゲート型を適用)

Download sample

このサンプルでは、汎用デリゲート型である "Proc" C# を宣言します。J# から汎用デリゲート型をインスタンス化し、適用します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

- [デバッグ] メニューの [デバッグ開始] をクリックするか、または F5 キーを押します。

  または

  コマンド ラインで、「**app**」と入力します。

参照

その他の技術情報

ジェネリックのサンプル (ジェネリックの使用)
Visual J# の複数の言語を含むサンプル
Visual J# の技術サンプル

ジェネリック スタック インターフェイスのサンプル (インターフェイスを利用)

Download sample

このサンプルでは、ジェネリック スタック クラスを使用する例を示します。ジェネリック スタック クラスはジェネリック スタック インターフェイスを実装します。J# から、このジェネリック スタック クラスをジェネリック スタック インターフェイスという概念で使用します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

- [デバッグ] メニューの [デバッグ開始] をクリックするか、または F5 キーを押します。

  または

  コマンド ラインで、「**app**」と入力します。

参照

その他の技術情報

ジェネリックのサンプル (ジェネリックの使用)
Visual J# の複数の言語を含むサンプル
Visual J# の技術サンプル

出力のジェネリック メソッドのサンプル (型の配列を出力)

Download sample

このサンプルの "プリンタ" 型には型の配列をコンソールに出力できるジェネリック メソッドがあります。このメソッドは、J# から使用され、さまざまな構成された型を提供します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドして実行するには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

- [デバッグ] メニューの [デバッグ開始] をクリックするか、または F5 キーを押します。

  または

  コマンド ラインで、「**app**」と入力します。

参照

その他の技術情報

ジェネリックのサンプル (ジェネリックの使用)
Visual J# の複数の言語を含むサンプル
Visual J# の技術サンプル

JavaCallsCOM_NET サンプル (.NET Framework 相互運用機能を使ったマネージ コンポーネントへの COM オブジェクトの公開)

Download sample

このサンプルでは、.NET Framework の COM 相互運用セマンティクスを使って、Visual J# アプリケーションから COM DLL にアクセスする方法を示します。Visual J# アプリケーションを新しく記述するときは、COM 相互運用の方法として .NET Framework セマンティクスを使うことが推奨されています。.NET Framework ツールの TlbImp によって、COM DLL に対するマネージ ラッパーが生成されます。このマネージ ラッパーは、Visual J# プログラムにインポートされます。適切なラッパーに対して **new** 演算子を使うと、COM コクラスのインスタンスが作成されます。開発者は、ラッパーのメソッドを呼び出すことにより、COM DLL のメソッドを呼び出すことができます。JActiveX® ラッパーを使った相互運用とは異なり、TlbImp によって生成されるラッパーは System.Object から拡張され、パラメータおよび戻り型として .NET Framework データ型を使用します。

.NET セマンティクスを使った COM 相互運用の詳細については、「COM への .NET Framework コンポーネントの公開」を参照してください。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

  ビルド済みの COM DLL がサンプルにインクルードされます。COM サーバーを再度ビルドしない場合は、既存の DLL を使う前に、「**regsvr32 COMServer.dll**」と入力して DLL を登録する必要があります。COM サーバーを再度ビルドする場合は、ビルド処理によって登録が行われるため、明示的に登録する必要はありません。

このサンプルを実行するには

- 開発環境では、\Client ディレクトリの Client.vjsproj ファイルを開き、F5 キーを押して、サンプルをビルドおよび実行します。

  または

コマンド ラインで、「**Client.exe**」と入力します。

参照

その他の技術情報

Visual J# の技術サンプル

JavaCallsCOM サンプル (Visual J++ 6.0 Java 言語/COM コンポーネントのアップグレード)

Download sample

このサンプルでは、Visual J++ 6.0 の Java 言語/COM アプリケーションを Visual J# にアップグレードする方法を示します。このサンプルの Clients サブディレクトリには、3 つのディレクトリが含まれています。SimpleClient ディレクトリのクライアントは、COM DLL にアクセスする単純なクライアントのアップグレードを示します。JActiveX® で生成されたラッパーもアップグレードされます。InterfaceMarshaling サンプルと MessagePump サンプルでは、アプリケーションを Visual J# にアップグレードするときに開発者が直面する可能性のある、スレッド処理に関連した特定の問題を回避する方法について説明します。

Visual J++ 6.0 の Java 言語/COM アプリケーションから Visual J# へのアップグレードの詳細については、「COM コンポーネントにアクセスする Visual J++ 6.0 アプリケーションのアップグレード」を参照してください。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

開発環境でサンプルをビルドするには

1. [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

   または

   コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

2. 開発環境で適切なプロジェクト ファイル (SimpleClient.vjsproj、InterfaceMarshaling.vjsproj、または MessagePump.vjsproj) を開き、ビルドします。

コマンド ラインでサンプルをビルドするには

- サンプルのルート ディレクトリで、「**BUILDALL.bat**」と入力します。

  サンプルに COM DLL がインクルードされます。COM サーバーを再度ビルドしない場合は、既存の DLL を使用できるように、「**regsvr32 COMServer.dll**」と入力して DLL を登録する必要があります。COM サーバーを再度ビルドする場合は、ビルド処理によって登録が行われるため、明示的に登録する必要はありません。

このサンプルを実行するには

- 開発環境では、F5 キーを押して、サンプルをビルドおよび実行します。

  または

  コマンド ラインでは、実行するクライアントのディレクトリから「**Client.exe**」と入力します。

参照

その他の技術情報

Visual J# の技術サンプル

Minesweeper サンプル (マインスイーパ ゲームのビルド)

Download sample

このサンプルでは、.NET Framework の次の機能をアプリケーションで使う方法を示します。

- イベント
- デリゲート
- 値型
- プロパティ
- 属性

次の名前空間の .NET Framework クラスが使用されます。

- System
- System.Drawing
- System.Windows.Forms
- System.ComponentModel

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- 開発環境では、F5 キーを押します。

  または

  コマンド ラインで、「**Minesweeper.exe**」と入力します。

参照

その他の技術情報

Visual J# の技術サンプル

PInvoke サンプル (アンマネージ API の呼び出し)

Download sample

このサンプルでは、共通言語ランタイムに用意されているプラットフォーム呼び出しサービスを使って、Visual J# でアンマネージ API を呼び出す方法を示します。プラットフォーム呼び出しは J/Direct® テクノロジに代わるものですが、J/Direct のように @dll() ディレクティブや Java 言語の型を使うのではなく、.NET Framework 相互運用属性と .NET Framework データ型を使ってアンマネージ API を呼び出します。Visual J# では、J/Direct も完全にサポートされています。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

サンプルを実行するには

- 開発環境では、F5 キーを押して、サンプルをビルドおよび実行します。

  または

  コマンド ラインで、「**PInvokeSample.exe**」と入力します。

コマンド ライン要件

Visual J# サンプルをコマンド ラインでビルドして実行する場合、サンプルが正しく機能するようにするには、vsvars.bat (<%Program Files%>\Microsoft Visual Studio 8\Common7\Tools ディレクトリ) を実行する必要があります。また、Visual Studio コマンド プロンプトを使用することもできます。コマンド プロンプトを使用するには、[スタート] ボタンをクリックし、[プログラム] をポイントします。次に、[Microsoft Visual Studio] をポイントし、[Visual Studio Tools] をクリックします。

参照

その他の技術情報

Visual J# の技術サンプル

RemotingSample サンプル (リモート処理サービスの作成)

Download sample

このサンプルでは、Visual J# を使ってリモート処理サービスを作成する方法を示します。この単純なサービスは、パラメータとして渡された文字列を操作し、結果を呼び出し元に返します。クライアントから呼び出す前に、サービスをホストする必要があることに注意してください。サービスをホストするには、サービスを DLL としてコンパイルし、IIS に登録します。

サンプルのディレクトリ構造を次に示します。

\bin

IIS を使ってリモート処理サービスにアクセスするときに必要な HelloService.dll を格納します。このサービスは、Clients\ConsoleClient ディレクトリおよび Clients\WebClient ディレクトリのクライアントから呼び出す前に、IIS に登録する必要があります。

\Service

HelloService.dll のソース コードおよび関連付けられた build.bat が格納されています。Build.bat は、HelloService.dll を作成し、bin ディレクトリにコピーします。

\Client\ConsoleClient

クライアントおよび関連付けられた build.bat が格納されています。コンパイラは、クライアントのコンパイル時にサービスのメタデータを参照する必要があります。このため、bin ディレクトリから HelloService.dll がコピーされます。クライアントを実行する前に、サービスを IIS に登録する必要があります。

\Client\WebClient

Web ブラウザからサービスにアクセスするために必要な .config ファイルと .aspx ファイルが格納されています。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：

詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

サービスを IIS に登録するには

1. サービスを DLL にコンパイルし、その DLL をサービスの読み込み元となる bin ディレクトリにコピーします。このサンプルでは、サンプル ディレクトリの下に bin サブディレクトリを作成し、そこに HelloService.dll をコピーします。build.bat を実行してサンプルをビルドすると、この処理が行われます。
2. インターネット サービス マネージャを起動し、サーバーのノードの下にある [既定の Web サイト] をクリックします。インターネット サービス マネージャを起動するには、[スタート] ボタンをクリックし、[プログラム] をポイントします。次に、[管理ツール] をポイントし、[インターネット サービス マネージャ] をクリックします。
3. [操作] メニューの [新規作成] をポイントし、[仮想ディレクトリ] をクリックします。
4. [次へ] をクリックして続行します。
5. エイリアスとして「**HelloService**」と入力し、[次へ] をクリックします。
6. サービスのあるディレクトリを入力します。手順 1. で DLL をコピーした bin ディレクトリまでの完全パスを入力します。ただし、bin ディレクトリを除きます。たとえば、「C:\Samples\RemotingService」と入力します。
7. [次へ] をクリックします。表示された既定の設定を受け入れ、もう一度 [次へ] をクリックします。[完了] をクリックします。

   これでリモート処理サービスが登録され、[インターネット サービス マネージャ] ウィンドウの [既定の Web サイト] の一覧に表示されます。
8. サービスを登録したディレクトリに web.config ファイルがあることを確認します。このファイルは、サービスについて記述するファイルです。ポート 80 の HTTP チャネルを使い、ブラウザまたはクライアントをとおしてクライアントからサービスにアクセスしようとすると、このファイルが自動的に読み込まれます。

このサンプルを実行するには

1. HelloService が IIS に登録されていることを確認し、Clients\ConsoleClient ディレクトリの実行可能ファイルを実行します。
2. HelloService が IIS に登録されていることを確認し、http://localhost/HelloService/Clients/WebClient に接続して、ブラウザからサンプルを実行します。

このサンプルを正しく動作させるために、Visual J# のインストール後にコンピュータの再起動を要求されることがあります。

参照

その他の技術情報

Visual J# の技術サンプル

Scoping サンプル (/securescoping の使用)

Download sample

このサンプルでは、vjc.exe に用意されている 2 つのスコープ指定オプション (/securescoping) の使用例を示します。スコープ指定が異なる 2 つのアプリケーションを Visual J# でコンパイルして実行し、メンバのスコープ指定のセマンティクスが package と protected で異なる場合の違いを示します。これらの違いは、パッケージが 2 つ以上のアセンブリにコンパイルされる場合に影響します。Visual J# で記述されたコンポーネントを .NET の他の言語に公開する場合には、これらの違いがより重要な意味を持ちます。保護されたスコープのオプションの詳細については、「/securescoping (パッケージ スコープを持つメンバへのアセンブリ外からのアクセスの禁止)」を参照してください。

サンプルでは、1 つのアセンブリ (.EXE) に含まれている package スコープのクラスから、同じパッケージの別のアセンブリ (.DLL) に含まれている package スコープのクラスへのアクセスを試みます。/securescoping オプションを指定せずに DLL をコンパイルした場合は、.EXE 内のクラスから DLL 内の package スコープのクラスにアクセスできます。/securescoping オプションを指定してコンパイルした場合は、java.lang.IllegalAccessError が発生します。

セキュリティに関するメモ：
このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

メモ：
詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- 開発環境では、F5 キーを押します。

  または

  コマンド ラインで「**PublicClass.exe**」と入力します。

コマンド ライン要件

Visual J# サンプルをコマンド ラインでビルドして実行する場合、サンプルが正しく機能するようにするには、vsvars.bat (<%Program Files%>\Microsoft Visual Studio 8\Common7\Tools ディレクトリ) を実行する必要があります。また、Visual Studio コマンド プロンプトを使用することもできます。コマンド プロンプトを使用するには、[スタート] ボタンをクリックし、[プログラム] をポイントします。次に、[Microsoft Visual Studio] をポイントし、[Visual Studio Tools] をクリックします。

参照

その他の技術情報

Visual J# の技術サンプル

SecurityProviders サンプル (サードパーティのセキュリティ プロバイダの指定)

Download sample

このサンプルでは、キーおよび署名の生成などの暗号操作を実行するためにサードパーティのセキュリティ プロバイダを使う必要がある場合に、vjsharp.config ファイルを使ってセキュリティ プロバイダを指定する方法を示します。既定では、vjsharp.config ファイルはインストールされません。Visual J# クラス ライブラリに付属する既定のセキュリティ プロバイダではなく、サードパーティのセキュリティ プロバイダを使って暗号操作を実行するには、このファイルを作成し、<%windir%>\Microsoft.NET\Framework\v<%version%> ディレクトリにコピーする必要があります。ファイル形式に関する情報と、このファイルを使ってセキュリティ プロバイダを読み込む方法については、「サードパーティのセキュリティ プロバイダを使うアプリケーションのアップグレード」を参照してください。

このサンプルでは、"*Sample*" アルゴリズムを使ったメッセージ ダイジェストの実装を提供するダミーのセキュリティ プロバイダを作成します。次に、vjsharp.config ファイルを作成して、"*Sample*" アルゴリズムを使って実装されたメッセージ ダイジェスト クラスの検索時に、このプロバイダを考慮に入れる必要があることを指定します。アプリケーションは、**java.security.MessageDigest** クラスの getInstance API を使って、ダミーのセキュリティ プロバイダからクラスを読み込もうとします。

セキュリティに関するメモ：

このサンプル コードは概念を示す目的で提供されているものです。必ずしも最も安全なコーディング手法に従っているわけではないので、アプリケーションまたは Web サイトでは使用しないでください。Microsoft は、サンプル コードが意図しない目的で使用された場合に、付随的または間接的な損害について責任を負いません。

サンプルのビルドと実行

ソリューション エクスプローラでサンプル ファイルを開くには

1. [サンプルのダウンロード] をクリックします。

   [ファイルのダウンロード] メッセージ ボックスが表示されます。

2. [開く] をクリックし、zip フォルダ ウィンドウの左列で、[ファイルをすべて展開] をクリックします。

   抽出ウィザードが開きます。

3. [次へ] をクリックします。ファイルを抽出するディレクトリを必要に応じて変更し、[次へ] をクリックします。

   [展開されたファイルを表示する] チェック ボックスがオンになっていることを確認して [完了] をクリックします。

4. サンプルの .sln ファイルをダブルクリックします。

   サンプル ソリューションがソリューション エクスプローラに表示されます。場合によっては、ソリューションの位置が信頼されていないという、セキュリティ上の警告が表示されることがあります。[OK] をクリックして続行します。

このサンプルをビルドするには

- [ビルド] メニューの [ソリューションのビルド] をクリックするか、または Ctrl キーと Shift キーを押しながら B キーを押します。

  または

  コマンド ラインで、「**BUILD.bat**」と入力します。

  メモ：

  詳細については、「コマンド ラインからのビルド (Visual J#)」を参照してください。

このサンプルを実行するには

- コマンド ラインで「**SecurityProvider.exe**」と入力します。

メモ：

サンプルを実行する前に、関連付けられている vjsharp.config ファイルを、サンプルを実行している vjslib.dll の適切なアセンブリ バージョンと公開キー トークンで更新してください。

参照

その他の技術情報

Visual J# の技術サンプル