您的潜力，我们的动力

Microsoft
微软（中国）有限公司

视频游戏开发系列课程(2)：
——基础编程概念及C#简介

讲师：俞晖

微软（中国）有限公司

MSDN Online Manager

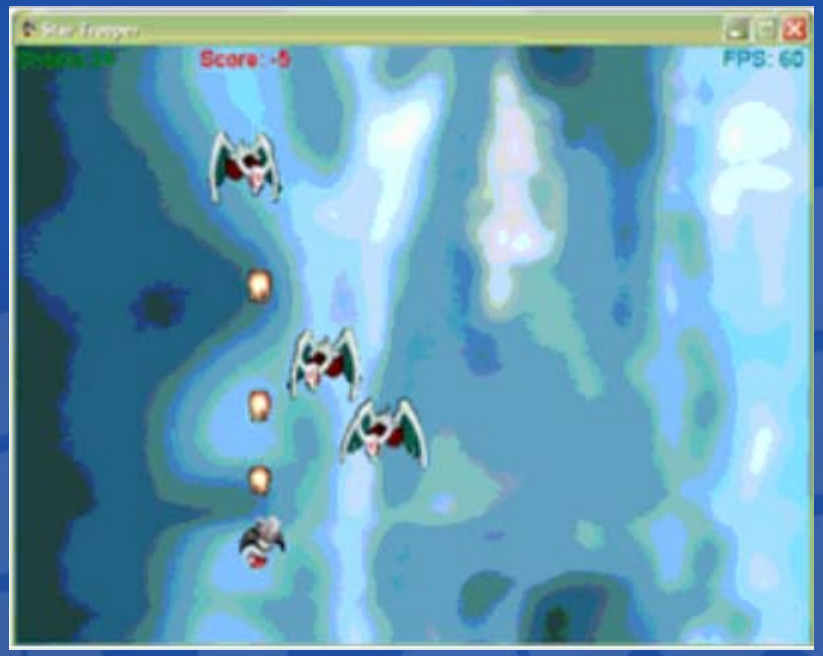
dolphin@vip.163.com

msdn

MSDN Webcasts

回顾

- 在第一节课上，您已经了解到了本次系列课程要为您讲述的知识概述，以及今后所有课程的概要信息
- Microsoft Visual C# Express 版本



本次课程内容包括

- C# 语言编程概念简述
- 变量、表达式，声明，操作符，函数，类，数组，结构和C#类型，构造器，属性器
- Direct X 名称空间简介（Optional）

本次课程需具备的条件

您的潜力，我们的动力

Microsoft
微软（中国）有限公司

- 不需要任何编程经验
- 在网站上下载相应的演示程序
- 安装Visual C# 2005 Express
- DirectX 9.0 SDK Update - (April 2005)
- DirectX 9.0 SDK Update - (Oct. 2005) Optional

Level 100

类(Classes)

- 类定义了一个“模板”，通过这个“模板”可以生成一个或多个对象
- 类定义了一个对象的“状态”和“行为”

```
public class BankCustomer
{
    .
    .    // first name, last name, balance,
    .


    .
    .    // deposit, withdrawal, send
monthly statement, etc.
    .
}
```

```
public class Text2D
  {
     private Font m_Font;
     public Text2D(Font font)
     {
        m_Font = font;
     }
  }
```

类的成员

- 在最简单的设计中，我们定义两种类的成员：
  - 属性：用于描述“状态”，保持“状态”
  - 方法：用于定义行为

```
public class BankCustomer
{
   public string   FirstName;    // fields/attributes
   public string   LastName;
   public decimal  Balance;

   public void Deposit(decimal amount)   // method
   {
      this.Balance += amount;
   }
}
```

实例化(Instantiation)

- 实例化就是创建对象的过程
- 使用new关键字来创建

```
BankCustomer  c;
c = new BankCustomer();
c.FirstName = "Kim";
c.LastName  = "Lee";
c.Balance   = 0.0M;

c.Deposit(1000.00M);
```


```
public class BankCustomer
{
   public string   FirstName;
   public string   LastName;
   public decimal  Balance;

   public void Deposit(decimal amount)
   {
      this.Balance += amount;
   }
}
```

对象 vs. 对象引用

- 对象变量保存的仅仅是对象的一个引用，不是对象本身！
- 这点是非常重要的！有助于理解.NET

```
BankCustomer  c;
c = new BankCustomer();
.
.
.

c = new BankCustomer();
.
.
.
```

静态(static)类成员

- 静态(static)中的成员不属于任何对象
  - 它可以通过类名直接访问
  - 实例成员必须通过建立对象的引用来访问

```
public class Math
{
   public const double PI = 3.14159;   // constants are implicitly static

   public static double Pow(double x, double y)
   {...}

   .
   .
   .
```

```
// compute area of circle...
area = Math.PI * Math.Pow(radius, 2.0);
```

封装对象的原理

- 什么是封装 (encapsulation)
- 封装的好处
  - 良好的封装能够减少耦合
  - 类内部的实现可以自由地修改
  - 类具有清晰的对外接口

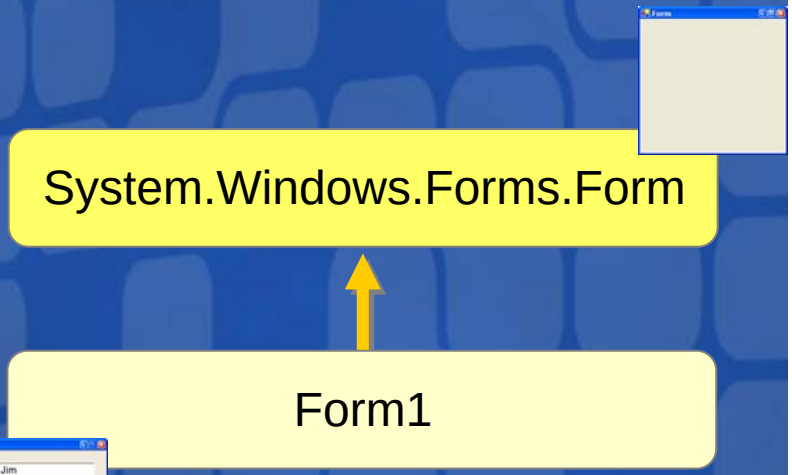
数据隐藏

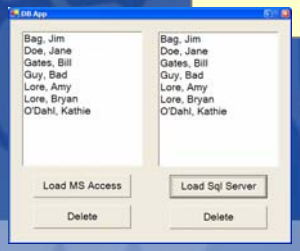
- 封装性最有用的方式之一
- 实现方法——访问限制修饰符
  - public 无限制，允许任何人来访问
  - protected internal = protected + internal
  - Internal 允许项目或程序内部的类来访问
  - protected 继承时子类可以对基类有完全访问权
  - private 只允许同一个类中的成员访问
- 属性和索引器也用来封装类的细节，并提供公用接口给该类的用户

继承性（inheritance）

- 一个类可以有能力直接从另一个类获得其代码和数据
- 派生类从基类那里获得其所有的成员
- 例：

  GUI设计的Form look and feel
  都是从.NET Form class
  继承下来的





如何实现继承?

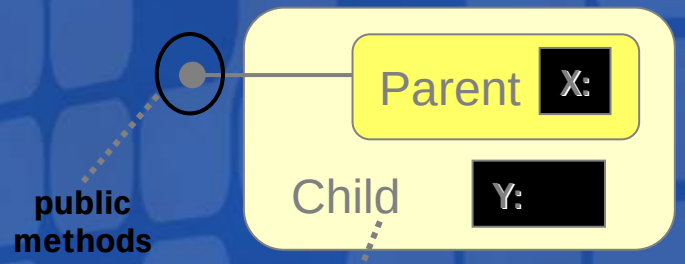
- public class GameWindow : Form
- C# 中只支持单继承
- 防止继承
  - public sealed class XX

继承透视图

- 如果Class Child是继承Class Parent而来
  - Child的对象中包含一个Parent的对象

```
public class Parent
{
    public int X;
    .
    .
}
```

```
public class Child : Parent
{
    public int Y;
    .
    .
}
```




```
// client...
Child  obj;
obj = new Child();
obj.X = 10;
obj.Y = 20;
```

如何访问基类成员？

- 派生类可以调用基类的方法
  - 通过使用base关键字
  - 派生类在访问基类的时候有一定的限制，不能访问 private 的成员；internal的基类成员只能被同一个程序集中的派生类访问

```
public class Account
{
        public double balance;
        public bool Withdraw(double amt)
        {
            balance -= amt;
            return true;
        }
}
```

```
public class CheckAccount:Account
{
        public bool Withdraw(double amt)
        {
            if(amt<= base.balance)
                return base.Withdraw(amt);
            else
                return false;
        }
}
```

对象引用

- CheckAccount ca = new CheckAccount();
- Account acc = new CheckAccount(); ???
- CheckAccount ca2 = acc ???

Solution: 数据类型转换

- Account acc = new CheckAccount()

  CheckAccount chk = acc; //Error

  CheckAccount chk = (CheckAccount) acc; //OK

- 如果acc不是CheckAccount的对象，那么运行时就将抛出 System.InvalidCastException

何时使用继承？

- 代码重用，减少编写的代码量
- 设计重用
  - 公用的字段和方法可以放到父类中，然后由其派生新的子类，之类有自己的字段和方法。
- 经验而言——”is a”的关系：
  - 如果A类是从B类中继承而来，即A是B的子类，那么我们可以说：“class A is-a class B”

多态性（1）

- 今天你多态了吗？
- 面向对象程序设计中的重要概念多态性。
  - 在运行时，可以通过指向基类的引用，来调用实现派生类中的方法。
  - 同一操作作用于不同的对象，可以有不同的解释，产生不同的执行结果，这就是多态性。
  - 多态性通过派生类覆写基类中的虚函数型方法来实现。

多态性（2）

- 编译时的多态性
  - 编译时的多态性是通过重载来实现的。对于非虚的成员来说，系统在编译时，根据传递的参数、返回的类型等信息决定实现何种操作。
- 运行时的多态性
  - 运行时的多态性就是指直到系统运行时，才根据实际情况决定实现何种操作。
  - C#中，运行时的多态性通过覆写虚成员实现。

重载（Overload）

- 重载—类中定义的方法可能有不同的版本
  - public bool Withdraw(double amt, string name)
    public double Withdraw(double amt)
- 特点（两必须一可以）
  - 方法名必须相同
  - 参数列表必须不相同
  - 返回值类型可以不相同

覆写（Override）

- 子类中为满足自己的需要来重复定义某个方法的不同实现——覆写
- 通过使用关键字override来覆写
  - public override bool Withdraw(…)
- 只有虚方法和抽象方法才能被覆写
- 要求：（三相同）
  - 相同的方法名称
  - 相同的参数列表
  - 相同的返回值类型

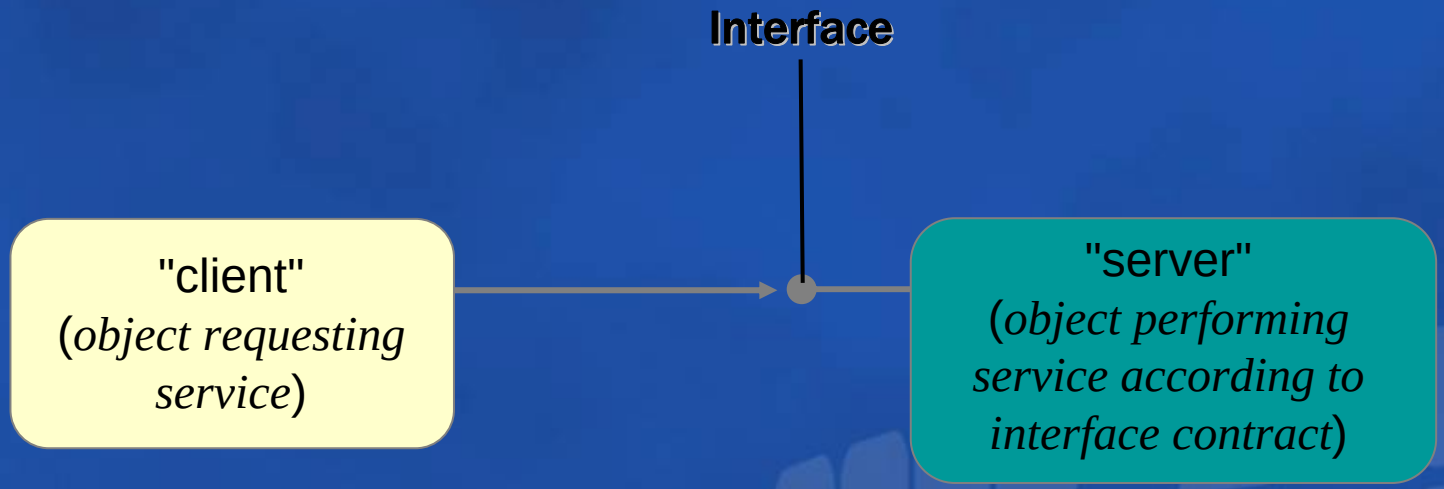
Override & Overload比较

Items	Override 覆写	Overload 重载
位置	存在于继承关系的类中	存在于同一类中
方法名	相同	相同
参数列表	相同	必须不同
返回值	相同	可以不相同

接口（1）

- 接口为类提供了蓝图
- 接口只提供定义
- 实现接口的数据类型必须提供接口成员的实现
- 接口本身可以从多个基接口派生

接口（2）



- “client” uses interface to communicate with server
- “server” implements interface to fulfill contract

接口（3）

- 使用interface关键字

```
public interface IAccount
{
     void Method1();
     string Method2(int x,
     int y);
     .
}
```

```
Public class InsuranceAccount :
     IAccount
{
     void Method1()
     {
        …
     }
     string Method2(int x, int y)
     {
        …
     }
}
```

构造函数 Constructor （1）

- 类中特殊的方法，多用于初始化实例的数据成员，在实例化 new 时被自动调用
- 函数名与类名总是相同，没有返回值

```
BankCustomer  c;
c = new BankCustomer();

c = new BankCustomer("kim", "lee", 1000.0M);
```

X

```
public class BankCustomer
{
    private string   FirstName;
    private string   LastName;
    private decimal  Balance;
    .
    .
    .

    public BankCustomer(string fn, string ln, decimal bal)
    {
        this.FirstName = fn;
        this.LastName  = ln;
        this.Balance   = bal;
    }
}
```

构造函数 Constructor （2）

- 通过重载，构造函数可以任意多

```
public class BankCustomer
{

    public BankCustomer(string fn, string ln, decimal bal)
   {
      this.FirstName = fn; this.LastName = ln; this.Balance = bal;
   }

   public BankCustomer ()
   {
      this.FirstName = “”; this.LastName = “”; this.Balance = 0.0M;
   }

   public BankCustomer(string fn, string ln)  // start with 0.0 balance
   {
      this.FirstName = fn; this.LastName = ln; this.Balance = 0.0M;
   }

   public BankCustomer(System.IO.StreamReader reader)  // initialize via stream (e.g. file)
   {
      this.FirstName = reader.ReadLine();
      this.LastName  = reader.ReadLine();
      this.Balance   = decimal.Parse( reader.ReadLine() );
   }
```

构造函数 Constructor（3）

- 类中特殊的方法，多用于初始化实例的数据成员，在实例化 new 时被自动调用
  - 函数名与类名总是相同
  - 没有返回值
  - 任意数量（通过重载Overload）
  - 构造函数间的调用
  - 访问修饰符的作用（public protected private）
- 如果没有显式定义，那么系统提供一个不带任何参数的public的构造函数
- 静态构造函数

属性 Property（1）

- 内部看像函数，外部看像字段
- 读/写由get/set存取器来进行
  - 没有参数列表
- Adv: 可以像给公有字段赋值一定轻松，同时允许通过get/set来控制对属性的访问

属性 Property （2）


```
bc.Balance = bc.Balance – amt;
```

```
public class BankCustomer
{
   .
   .
   .
   private decimal  m_Balance;   // field
   public  decimal  Balance      // property
   {
      get { return this.m_Balance; }
      set
      {
         if (value < 0.0M)   // new balance cannot be negative!  So ignore…
           ;
         else   // update balance to new value
           this.m_Balance = value;
      }
   }
```

Struct 结构

- Struct 也称为值类型，Class也称为引用类型
- Struct 不允许无参数的构造器
- Struct 不会被系统自动调用构造器

```
struct Point
{
    public int x;
    public int y;

    public Point(int x1, int y1)
    {
        x=x1;
        y=y1;
    }
}

class Rectangle
{
    public Point top,bottom;

    public Rectangle()
    {
      top = new Point(1,2);
      bottom = new Point(100,200);
    }
}
```

名称空间

- 名称空间可以按逻辑对类进行划分
  - 增强可读性（ *System.Windows.Forms.Form* ）
  - *减少了类名的冲突（my classes &your classes)*
- *例如：*

```
namespace N
{
   public class Class1
   {
      .
      .
      .
   }
   public class Class2
   {
      .
      .
      .
   }
}
```

N.Class1

N.Class2

完全引用方式

- 从名称空间的最外层完全引用
  - 优势？
  - 劣势？

```
TeamX.BusinessTier.Customer  c;
c = new TeamX.BusinessTier.Customer(...);
   .
   .
   .
```

导入(importing)

- 你可以导入名称空间
  - 使用using关键字在文件的顶部导入要使用的名称空间

```
using TeamX.BusinessTier;
   .
   .
   .

Customer  c;
c = new Customer(...);
```

导入“别名”(alias)

- 过度使用using将带来问题
  - 干扰可读性
  - 增加名称冲突的可能性
- 使用alias是一种折中的方式

```
using BT = TeamX.BusinessTier;
   .
   .
   .

BT.Customer  c;
c = new BT.Customer(...);
```

Direct X 名称空间（1）

- **Microsoft.DirectX.AudioVideoPlayback**
- The AudioVideoPlayback application programming interface (API) provides for basic playback and simple control of audio and video files.

- **Microsoft.DirectX.Direct3D**
- Microsoft Direct3D is a low-level graphics application programming interface (API) that enables you to manipulate visual models of 3-D objects and take advantage of hardware acceleration.

Direct X 名称空间（2）

- **Microsoft.DirectX.DirectInput**
- Microsoft DirectInput is used to process data from a keyboard, mouse, joystick, or other game controller.

- **Microsoft.DirectX.DirectSound**
- Microsoft DirectSound provides a system to capture sounds from input devices and play sounds through various playback devices using advanced 3-D positioning effects, and filters for echo, distortion, reverberation, and other effects.

Direct X 名称空间（3）

- **Microsoft.DirectX.PrivateImplementationDetails**
- The PrivateImplementationDetails namespace contains structures and interfaces that allow the managed application programming interface (API) to have access to the unmanaged portions of the Microsoft DirectX API.

- **Microsoft.DirectX.Security**
- Controls permissions related to Microsoft Direct3D, Microsoft DirectInput, Microsoft DirectPlay, and Microsoft DirectSound.

DEMO

DirectX Tutorials

软件支持

- DirectX 9.0 SDK Update - (April 2005)

http://www.microsoft.com/downloads/details.aspx?FamilyId=AFC15F29-D7C9-4CF7-A8D5-8AB81F14AE1B&displaylang=en

- DirectX 9.0 SDK Update - (October 2005)

http://www.microsoft.com/downloads/details.aspx?FamilyID=1c8dc451-2dbe-4ecc-8c57-c52eea50c20a&displaylang=en

- Visual C# 2005 Express

http://msdn.microsoft.com/vstudio/express/visualCsharp/default.aspx

http://www.microsoft.com/china/msdn/express/csharp.aspx

(Available in next week. 11-14 2005)

获取更多MSDN资源

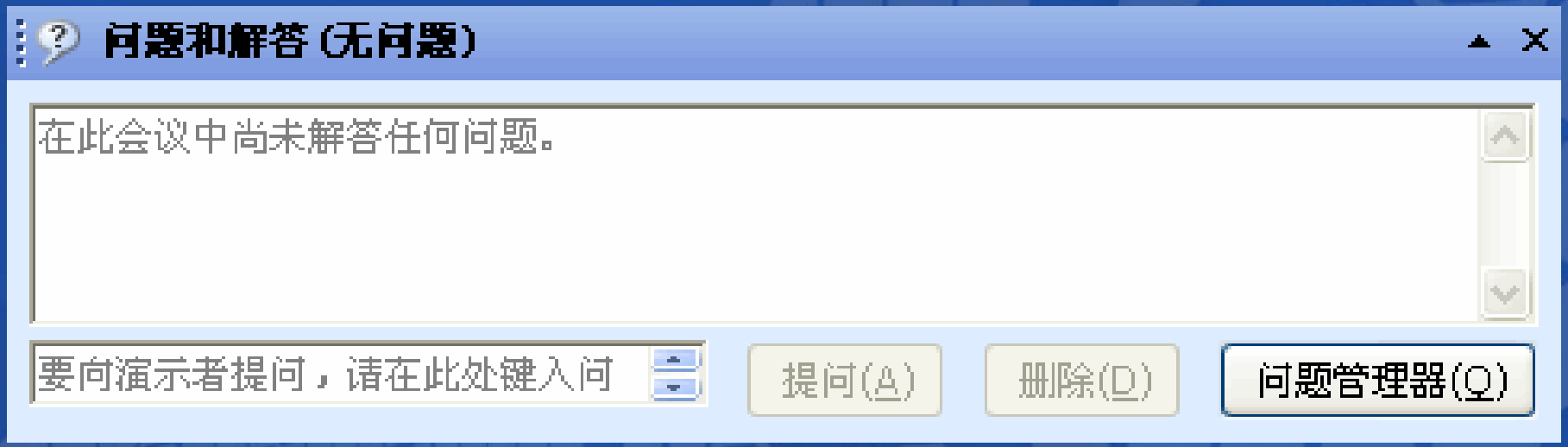
您的潜力，我们的动力

Microsoft
微软（中国）有限公司

- 微软学生中心
  http://www.msuniversity.edu.cn
- **MSDN**中文网站
  http://www.microsoft.com/china/msdn
- **MSDN**中文网络广播
  http://www.msdnwebcast.com.cn
- **MSDN Flash**
  http://www.microsoft.com/china/newsletter/case/msdn.aspx
- **MSDN**开发中心
  http://www.microsoft.com/china/msdn/DeveloperCenter/default.mspx

Question & Answer

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。




您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts