

【ねらい】

クラスのメンバについて理解します。

メンバのアクセス修飾子および、静的メンバの使い方について理解します。

1 メンバの種類

クラスは、データやクラスの動作を定義する要素をその内部に持つことができます。それら要素はまとめて**メンバ**と呼ばれます。以前説明したフィールドやメソッドもメンバの一種です。クラスのメンバには以下の種類があり、クラスの中で定義することができます。

フィールド	クラスの一部と見なされるデータです。値の変更が可能です。
定数	初期化後に値が変更されることのないデータです。
メソッド	クラスが実行できる動作を定義します。引数を受け取り、戻り値を返すことができます。
コンストラクタ	クラスのインスタンスを生成する時に呼び出されるメソッドです。一般にデータの初期化のために使用します。
デストラクタ	インスタンスがメモリから削除される時に呼び出されるメソッドです。リソースの解放のために使用します。
プロパティ	メソッドの一種ですが、フィールドのようにアクセスすることができます。
イベント	ボタンのクリックやメソッドの終了などの、特定の状況の発生を他のオブジェクトに通知する手段に用います。
演算子	+ や * などの演算の動作を、クラス用に再定義します。
インデクサ	配列のようにインデックスを使用してデータにアクセスできるようにします。
クラスの中で宣言された型	通常このクラスの中だけで使用するオブジェクトを表すために使用します。

この後の節では、フィールド、定数、メソッド、コンストラクタ、デストラクタについて順に説明します。他のメンバについては今回のシリーズでは取り上げません。

2 メンバの定義

クラスのメンバは、クラスの中で定義します。次の例では `Train` クラスのメンバとして、`Express`

フィールドと、MoveNext メソッドを定義しています。

<pre>class Train { public bool Express; public void MoveNext() { // 省略 } }</pre>	<p>Train クラスの定義。</p> <p>Express フィールドの定義。</p> <p>MoveNext メソッドの定義。</p>
---	--

クラスの定義と同様に、メンバについても**アクセス修飾子**を付けて定義することで、アクセス制限を設定することができます。上の例では `public` がアクセス修飾子になります。`public` が指定されたメンバは、アクセス制限がなく、外の任意のクラスから呼び出すことができます。`public` の代わりに `private` を指定した場合には、メンバを定義したクラスの中だけからアクセスできるメンバになります。アクセス修飾子にはこの他に `protected` と `internal` がありますが、今回は扱いません。メンバの定義でアクセス修飾子を指定しなかった場合には既定で `private` になります。

外部からのアクセスを予定していないメンバは `private` にしておくことをお勧めします。これは後でそのメンバの名前や動作を変更したくなった場合などに、`private` のメンバであれば、そのクラスの中だけでコードを修正すれば良いことが確実になるからです。また、このクラスを利用してプログラムを作成する場合には、`private` のメンバについてはまったく気にせずに、`public` なメンバだけを使用してコーディングすれば良いことが明確になります。

3 静的メンバ

静的メンバは、クラスのインスタンスを作成せずにアクセスすることができるメンバです。静的メンバの例としては、`Console` クラスの `WriteLine` メソッドがあります。またプログラムの開始点である `Main` メソッドも静的メンバです。メンバを静的メンバにするには、`static` キーワードを付けてメンバを定義するようにします。

<pre>class Train { public static int MaxSpeed; public static void SetMaxSpeed(int speed) { MaxSpeed = speed; } }</pre>	<p>Train クラスの定義。</p> <p>静的フィールド MaxSpeed の定義。</p> <p>静的メソッド SetMaxSpeed の定義。</p> <p>パラメータ speed を MaxSpeed に代入。</p>
---	---

上の例では フィールド `MaxSpeed` とメソッド `SetMaxSpeed` を静的メンバとして定義しています。静的メンバはクラスの各インスタンスに共通のデータや動作を定義したい時などに使用します。列車の最高速度がすべての列車で同じであるならば、列車ごとに最高速度のフィールドを持つ必要はありません。最高速度を静的なフィールドとして定義しておけば、全ての列車に共通の統一された値として扱うことができ、メモリの節約にもなります。最高速度を変更する場合にも、各インスタンスのフィールドをそれぞれ変更する必要がなくなります。

静的メンバとして定義されているフィールド、メソッドはそれぞれ**静的フィールド**、**静的メソッド**と呼ばれます。静的でない通常のメンバ、フィールド、メソッドを特に区別して呼ぶ場合には、**インスタンスメンバ**、**インスタンスフィールド**、**インスタンスメソッド**という呼び方をします。

クラスのメンバが全て静的なメンバである場合、クラス自体を `static` を付けて宣言し、静的クラスにすることもできます。静的クラスにするとインスタンスが生成されないことが保証されます。`Console` クラスや、`Math` クラスなどは静的クラスとして定義されています。

4 メンバの利用

クラスの中で `public` キーワードを付けて定義されているメンバは、他のクラスからドット演算子を使用して参照することができます。インスタンスメンバの場合には、インスタンスを生成した後で、インスタンスに対してドット演算子を使用します。

```
Train train1 = new Train();
train1.Express = true;
train1.MoveNext();
```

`Train` クラスのインスタンスを生成して `train1` に代入。
`train1` の `Express` フィールドに `true` を代入。
`train1` の `MoveNext` メソッドの呼び出し。

静的メンバの場合には、インスタンスではなく、クラス名に対してドット演算子を使用します。

```
Train.SetMaxSpeed(100);
string output = "最高速度: " + Train.MaxSpeed;
```

`Train` クラスの静的メソッド `SetMaxSpeed` を呼ぶ。
`Train` クラスの静的フィールドの参照。