オブジェクト指向と
ゲームプログラミング

○　Java 基礎編 － 第２回　クラス　○

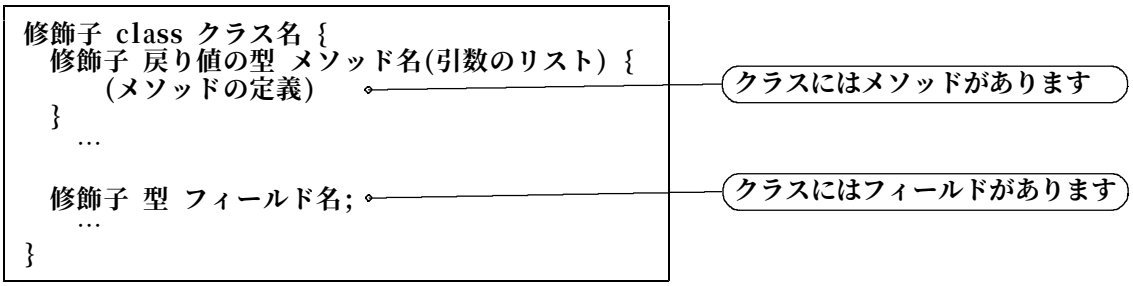
クラス

クラスとは、動作と属性をまとめたものに名前を付けたものです。オブジェクト指向設計で洗い出されたオブジェクトは、ソースコード上でクラスとして定義されます。オブジェクト指向プログラミングでは、クラスを定義することが基盤となっていますが、ソースコードを記述しながら動作と属性をまとめていくのではなく、設計の行程でオブジェクトすなわちクラスを洗い出すことが先決であり、クラスの動作と属性を洗い出すことが後になることに注意しましょう。

クラスは、オブジェクトを作り出す方法を定義したものであり、実体を持つわけではありません。オブジェクトは、クラスのインスタンス(実体)として作り出されます。

クラスの定義

クラスの定義には、クラスの属性となるフィールドの定義と、クラスの機能となるメソッドの定義があります。特殊なメソッドとしてコンストラクタやファイナライザを定義することもあります。クラス定義の書式は、以下のようになります。



▲クラスの宣言

クラスの定義は、classというキーワードの後にクラス名を指定し、その後にある{と}で囲まれたブロックの中に、クラスの持つフィールド、メソッド、およびコンストラクタを定義します。フィールドの定義とは、クラスが持つべき変数や定数を記述することです。メソッドの定義とは、メソッドのプロトタイプを示し、その処理内容を記述することです。

以下のクラスは、ゲームのキャラクターを表現するCharacterクラスです。ここでは、説明を簡単にするため、フィールドは、「ｘ座標」「ｙ座標」「アニメーション番号」で、メソッドは「移動」「アニメーション」だけにしています。コンストラクタはありません。継承、ポリモーフィズムも使っていません。

キャラクター
```
属性：
  ｘ座標(double)
  ｙ座標(double)
  アニメーション番号(int)

動作：
  移動
  アニメーション
  (属性の初期化)
```

Characterクラス
```
class Character {
    double   posX;          // ｘ座標
    double   posY;          // ｙ座標
    int      animeNo;       // アニメーション番号

    public void move(double addX, double addY) {
        // 移動処理(省略)
    }

    public void animation() {
        // アニメーション(省略)
    }
}
```

Characterクラスは、double型の変数posX, posYとint型の変数animeNoを属性として持っています。このようなクラス内部のデータがフィールドです。また、このクラスは、move, animationという２つの動作を持っています。これらをメソッドと呼びます。

アクセス修飾子

クラスおよびクラスのメンバには、アクセス指定を行います。アクセス指定には、キーワードpublic, protected, private, そしてアクセス修飾子なし(デフォルトアクセス)の４つあります。

publicアクセスは、公開してもよいことを示します。これを付けて宣言したクラスおよびメンバは、どこからでも自由に使うことができます。

protectedアクセスは、そのクラスとそれを継承したクラス、および同じパッケージ内の別のクラスから使うことができます。

privateアクセスは、そのクラスの中でのみ使うことができます。

デフォルトアクセスは、何も付けずに宣言した場合に適用されます。そのクラスの中と、同じパッケージの別のクラスから使うことができます。

カプセル化

クラスではメンバの公開および非公開が、バグを減らす重要な役割を果たすことになります。

public指定したメンバは「公開」となります。その意味は「それらのメンバはどこからでも直接アクセスできる」ということです。前述のCharacterクラスでは、moveメソッドとanimationメソッドがpublicであるため、ほかのクラスでも呼び出すことができます。

private指定したメンバは「非公開」になります。その意味は、クラスの内部でしかアクセスできないということです。前述のCharacterクラスでは、private指定されたメンバはありません。

アクセス修飾子を何も付けない場合、「デフォルトアクセス」になります。これは、そのクラスの中と、同じパッケージ内の別のクラスからしかアクセスできないというものです。前述のCharacterクラスでは、変数posX, posY, animeNoがこれにあたり、同じパッケージに属しないクラスからアクセスしようとすると、コンパイルエラーになります。

Characterクラスでは、フィールドがすべてデフォルトアクセスになっています。これは、Characterクラスを使うプログラマに「フィールドを外部から直接アクセスしてはならない」という制約を課していることになります。
なぜ、このように面倒なことをするのかというと、バグを減らすという意味があります。このような制約を課す方法は、不便に感じることも多々あります。しかし、メソッドをとおして間接的にしかアクセスさせないようにし、さらにエラーチェックを厳重にすることにより、「非現実的な操作」や「不正な操作」ができなくなり、バグを減らすことができます。フィールドを外部から直接変えられるようにしておくと、後々とんでもない数値を代入するといった操作から不具合が発生するかもしれません。クラスを設計する人とクラスを利用する人が異なる場合もあるからです。

このように、データを内部に置いて、外部から触れさせない仕組みをカプセル化(データ隠蔽または情報隠蔽)といいます。フィールドは、原則として非公開(public以外)にし、フィールドへの操作はメソッドをとおして間接的に行うように設計します。

また、すべてのメソッドを公開しなければならないということはありません。クラスの外部からまったく利用せず、クラスの内部だけで呼び出されるだけのメソッド(たとえば、クラス内の一時的な作業や演算を行うようなメソッド)をわざわざ公開する必要はありません。このような場合は非公開にし、隠蔽しておきます。

クラスを設計する段階で適切にアクセス指定をしておけば、ほかの人がそのクラスを利用しても、不具合の発生しにくいプログラムが作成できるわけです。

フィールドの定義

フィールドは、クラスの持つ属性であり、クラスの中でメソッドの外に記述された変数や定数として定義されます。フィールドは、そのクラスの持つメソッドが読み書きすることを想定しています。

フィールドの定義は、データ型を指定して変数や定数を宣言するだけです。C++と違い、宣言と同時に初期値を与えることができます。フィールドが定数の場合には、データ型の前にfinalを指定します。finalは値を変更できない変数(すなわち定数)であることを示し、初期値を設定した後は、いっさいの変更を行うことができません。

フィールドのデータ型に、別のクラスを指定することができます。この場合、別のクラスを利用していることになりますが、継承ではありません。変数のデータ型が別のクラスになっているだけです。

メソッドの定義

メソッドは、クラスの持つ動作であり、メソッドのプロトタイプと処理内容を記述します。メソッドは、クラスの中に、以下のように記述します。

```
class Character {
    public final static int   MAX_ANIME = 5;    // アニメーション最大数

    double      posX    = 0.0;                   // x座標
    double      posY    = 0.0;                   // y座標
    int         animeNo = 0;                     // アニメーション番号

    // 移動
    public void move(double addX, double addY) {
        posX += addX;
        posY += addY;
    }

    // アニメーション
    public void animetion() {
        animeNo++;
        if(MAX_ANIME <= animeNo)
            animeNo = 0;
    }

    // アクセスメソッド
    public double getX() { return posX; }        // x座標取得
    public double getY() { return posY; }        // y座標取得
}
```

練習問題

1　以下の文章の内容が正しい場合には○、間違っている場合には×を付けましょう。

(１)public指定されたメンバは、クラスの外からアクセスすることができる。
(２)private指定されたメンバは、クラスの外からアクセスすることができない。
(３)アクセス修飾子がないメンバは、どこからでもアクセスすることができる。
(４)メソッドの中で定義されている変数をフィールドと呼ぶ。
(５)すべてのフィールドは、オブジェクトが作成されていないとアクセスすることができない。
(６)フィールドを持たず、メソッドのみのクラスを定義することはできない。
(７)フィールドの定義でfinalを指定すると、定数のフィールドとなる。
(８)フィールドの型に、別のクラスを指定することができる。

2　クラスの定義において、フィールドを非公開にする理由を説明しましょう。

3　次のように整数値の座標を表すクラスPointを作成しましょう。ただし、座標の範囲はｘが０～６４０、ｙが０～４８０となるようにしましょう。

フィールド
　ｘ座標…posX
　ｙ座標…posY

メソッド

```
void setX(int px);          // ｘ座標を設定する
void setY(int py);          // ｙ座標を設定する
int getX();                 // ｘ座標を得る
int getY();                 // ｙ座標を得る
```