オブジェクト指向と ゲームプログラミング

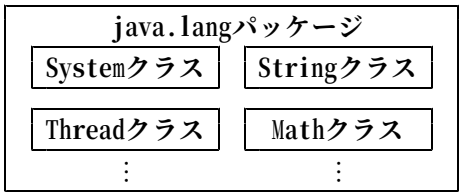
○　Java 基礎編 － 第１０回　パッケージ　○

パッケージ

プログラムが大規模になると「関連のあるクラスをまとめたい」という要求が生じます。また、複数のプログラマで作業を行っていると、しばしばクラス名の衝突が起きます。このような場合に用いるのがパッケージです。Javaでは、クラスファイルをパッケージという単位にまとめることができます。

クラス名が異なれば同じ名前のフィールドやメソッドを定義できるように、パッケージが異なれば、同じ名前のクラスやインタフェースを定義することができます。

独自に作成したクラスは、デフォルトで名前のないパッケージに含まれます。たとえば、独自に数学処理を行うMathというクラスを作成したとしても、Javaが提供するMathクラスとは、別のものと見なされます。Javaが提供するMathクラスは、java.langパッケージに含まれているからです。



▲パッケージには、クラスとインタフェースが含まれてます

パッケージ宣言

任意のパッケージにクラスを含るには、キーワードpackageでパッケージ宣言を行います。ソースファイルの先頭でパッケージ宣言を行うと、そのパッケージに属しているものとなり、同じ名前のパッケージがまとめられます。

たとえば、Testクラスをmypackパッケージに含めるには、以下のようになります。

```
package mypack;

public class Test {
    public static void main(String[] args) {
        System.out.println("KTU");
    }
}
```

こうすると、Testクラスは、mypackパッケージに含まれることになります。しかし、このままでは実行できず、エラーになってしまいます。パッケージに含まれるクラスファイルは、そのパッケージの名前と同じフォルダに入れなければならないのです。

上記のTestクラスの場合、mypackパッケージに含まれているので、カレントフォルダ(プロジェクトのフォルダ)の下にmypackというフォルダを作り、その中にクラスファイル(Test.class)を置きます。また、コマンドプロンプトから実行する場合も、以下のようにしなければなりません。

```
java mypack.Test
```

なお、ソールファイル(.java)もパッケージのフォルダに置いた方が、管理しやすくなります。

パッケージの階層化

パッケージは階層化できます。mypackパッケージの中でapplicationパッケージを作る場合、mypackフォルダの下にapplicationフォルダを作り、その中にクラスファイルを置きます。この場合は、パッケージ宣言は以下のようになります。

```
package mypack.application;
```

import

ほかのパッケージにあるクラスは、パッケージ名をつけて使用します。

```
// 異なるパッケージから、mypack.applicationパッケージのAppクラスを使う
class Game {
    mypack.application.App   appli = new mypack.application.App();
}
```

しかし、「mypack.application.App」のように、パッケージ名とクラス名をいちいち書くのはかなり面倒です。このような場合、import宣言をすればパッケージ名を省略できます(名前が重複しない場合)。

```
// 異なるパッケージから、mypack.applicationパッケージのAppクラスを使う
import mypack.application.App;

class Game {
    App   appli = App();
}
```

パッケージ内のすべてのクラスをインポートする場合は、「import mypack.application.*」のようにワイルドカード'*'を用います。ただし、サブパッケージまで効力は及びません。

なお、packageとimportを同時使用する場合は、packageを先に記述します。