Chapter 4

継承、インタフェース、パッケージ

以下の指示に従って、クラス、オブジェクトを作成・実行せよ。なお、文中では個別に指示しないが、変更を行ったら適宜コンパイルすること。

4.1 クラスの継承

- 次の Chokin クラスを作成せよ。（次以降の手順ではあらかじめコンパイルされている必要がある）

```
class Chokin{
        int okane=0;
        Chokin(){
           okane=10000;
        }
        void addOkane(int i){
           okane+=i;
        }
        void print(){
           System.out.println(okane+"yen");
        }
}
```

- Chokin クラスを継承し BankAccount クラスを作成せよ。ただし、BankAccount クラスには String 型の変数 bank（初期値は”TokyoBank”）がフィールドとして含まれることとし、print メソッドを、出力の中に okane+”yen”

だけでなく bank の値も含めるように再定義（オーバーライド）せよ。この段階では、BankAccount クラスのコンストラクタは作らなくてよい。

- Test04A クラスを作成し、次のような処理を行う main メソッドを作成し、実行せよ。なお、継承されたクラスのコンストラクタは自動的に親クラスのコンストラクタを処理の最初に呼び出すことを思い出せ。「BankAccount クラスのオブジェクト bank1 を作成する。次に、addOkane メソッドを利用して bank1 のお金を 5000 増やす。その後、print メソッドで bank1 の状態を画面に出力する。」

- BankAccount クラスのコンストラクタとして以下のものを追加せよ。

```
BankAccount(String bank){
   this.bank=bank;
}
```

- Test04A クラスの main メソッドを、bank1 オブジェクトの生成時に引数に”OsakaBank”を入れるコンストラクタを呼び出すように変更し実行せよ。この時コンストラクタがどのような順序で呼び出されたか考えてみよう。

4.2 インタフェース

- 次の二つのインタフェースを作成せよ。(Circle.java ファイルに二つのインタフェースの定義を記述しコンパイルする)

```
interface Circle{
 void setR(double r);
 double getArea();
}
interface Painted{
 boolean isFilled();
}
```

- インタフェース Circle と Painted を実装するクラス PaintedCircle を作成し、実行せよ。但し、以下のフィールド、メソッドを備えているものとする。

- フィールド double r=1.0 (半径に相当)
- メソッド public void setR(double r)　フィールド r に引数の値をセットする処理を行うものとする。
- メソッド public double getArea()　ただし、半径 r の値から円の面積を計算して返す処理を行うものとする。
- メソッド main　ただし、PaintedCircle クラスのオブジェクト pc を生成し、オブジェクト pc の setR メソッドを用いて、半径を 10.0 にセットし、さらに getArea メソッドを用いて面積を画面に表示する

実は、この仕様では PaintedCircle は正しく実行できない。その理由について考察せよ。

- クラス PaintedCircle にメソッド public boolean isFilled() を追加せよ。ただし、単に true を返す処理を行うものとする。再度 PaintedCircle を実行せよ。

4.3　パッケージ

以下、Linux 環境で実施しているものとする。

- まず、各自のホームディレクトリ直下に java0000 という名前のディレクトリを作成し、カレントディレクトリを java0000 に移動せよ。
- カレントディレクトリ直下に com という名前のディレクトリを作成せよ。次に、ディレクトリ com の中に shibaura という名前のディレクトリを作成し、さらにディレクトリ shibaura 内にディレクトリ tools およびディレクトリ system を作成せよ。
- 次のプログラムを Car.java として作成し、コンパイルして出来たファイル Car.class を上記のディレクトリ tools に配置せよ [1]。(クラスとメソッドに public を忘れずにつけること)

[1]実は、予めコンパイルせずともソースファイルをパッケージの対応するディレクトリに置いておけばこれを利用するクラスをコンパイルするときに同時にコンパイルはされるのだが、class ファイルがパッケージの該当部分に存在することが本質的に大事なので、本演習では予めコンパイルしてもらう。

```
package  com.shibaura.tools;

public class Car{
        double speed=0.0;
        public void setSpeed(double ds){
              speed=ds;
        }
        public double getSpeed(){
              return speed;
        }
}
```

- 次のプログラムを Car.java として作成し、コンパイルして出来たファイル Car.class を上記のディレクトリ system に配置せよ。(クラスとメソッドに public を忘れずにつけること)

```
package  com.shibaura.system;

public class Car{
        double speed=0.0;
        public void setSpeed(double ds){
              speed=ds;
        }
        public double getSpeed(){
             return speed*3600.0/1000.0; // converted to (km/h)
        }
}
```

- 次のプログラムを Test04C.java として作成し、ディレクトリ java0000 直下に置いてコンパイルし、実行せよ。また、com.shibaura.system パッケージの Car クラスを用いる場合はどうしたらよいか。適切に修正して実行せよ。

```
import com.shibaura.tools.*;

class Test04C{
```

```
    public static void main(String[] args){
            Car myCar = new Car();
            myCar.setSpeed(100.0);
           System.out.println("velocity:"+myCar.getSpeed());
    }
}
```