独習 Java（第3版）

6.7 変数の修飾子
6.8 コンストラクタの修飾子
6.9 メソッドの修飾子
6.10 Object クラスと Class クラス

6.7 変数の修飾子(1/3)

- 変数宣言の直前に指定できる修飾子
  - 全部で7種類ある

キーワード	意味
final	定数として使える変数
private	同じクラスのコードからしかアクセスできない変数
protected	サブクラスまたは同じパッケージ内のコードからしか アクセスできない変数
public	他のクラスからアクセスできる変数
static	インスタンス変数ではない変数
transient	クラスの永続的な状態の一部ではない変数
volatile	不意に値が変更されることがある変数

6.7 変数の修飾子(2/3)

- 一部の修飾子は同時に指定できない
  - public, protected, privateの3つのキーワードからは1つしか指定できない
- これら修飾子を一切指定しないと変数は非final、非transient, 非volatileなインスタンス変数と解釈される
- このような変数には同じパッケージ内のコードからしかアクセスできない

6.7 変数の修飾子(3/3)

- final変数の使い方の例

```
class L {
        static final int x = 5 ;
}
class StaticFinal {
        public static void main(String args[]) {
                System.out.println(L.x);
        }
}
```

  - 実行結果は“5”が出力される
- このプログラムではクラスLの中の変数xの値を変更することは出来ない

6.8 コンストラクタの修飾子(1/2)

- コンストラクタの宣言に指定できる修飾子
  - 全部で3種類ある

キーワード	意味
private	同じクラスのコードからしかアクセスできないコンストラクタ
protected	サブクラスまたは同じパッケージ内のコードからしか アクセスできないコンストラクタ
public	他のクラスからアクセスできるコンストラクタ

- 2つ以上同時に指定できない
- 指定がなければ同じパッケージ内の
  コードからしかアクセスできない

6.8 コンストラクタの修飾子(2/2)

- 例 :

```
class Test {
        int x ;
        public Test( int y ) {
                x = y ;
        }
        private Test() {
        }
}
class PrivateConstructor {
        public static void main( String args[] ) {
                Test t1 = new Test( 30 ) ;
                System.out.println( t1.x ) ;
                //Test t2 = new Test() ;
        }
}
```

実行結果は“30”が表示される

publicなので呼び出せる

privateなので呼び出せない

6.9 メソッドの修飾子(1/4)

- メソッドの宣言に指定できる修飾子
  - 全部で8種類ある

キーワード	意味
abstract	このクラスでは実装しないメソッド
final	オーバーライドできないメソッド
native	Javaのバイトコードではなく、 ホストCPUで使われるマシン語で実装されるメソッド
private	同じクラス内のコードからしか呼び出されないメソッド
protected	サブクラスまたは同じパッケージ内のコードからしか アクセスできないメソッド
public	他のクラスからアクセスできるメソッド
static	インスタンスメソッドではないメソッド
synchronized	実行を開始するときにロックを取得するメソッド

6.9 メソッドの修飾子(2/4)

- abstractメソッドを含むクラスはそれ自体もabstractで宣言されている必要がある
- public, protected, privateは他の修飾子と同時に指定できない
- synchronizedはマルチスレッドのプログラムを作成する場合に重要
- これらの修飾子を指定しないとメソッドは非abstract, 非final, 非native, 非synchronizedとなり、同じパッケージ内からしかアクセスできない

6.9 メソッドの修飾子(3/4)

- 例:

```
class Singleton {
        static Singleton singleton;
        private Singleton() {}
        public static Singleton getInstance() {
                if (singleton == null)
                singleton = new Singleton();
                return singleton;
        }
}
class SingletonDemo {
        public static void main(String args[]) {
                Singleton s1 = Singleton.getInstance();
                Singleton s2 = Singleton.getInstance();
                if (s1 == s2)
                        System.out.println("Equal");
                else
                        System.out.println("Not equal");
        }
}
```

singletonオブジェクトが
まだないときには作成

singletonオブジェクトを返す

singletonオブジェクトを
取得するために
getInstance()を使用している

6.9 メソッドの修飾子(4/4)

- 先ほどのプログラムの実行結果

  Equal

- これはs1とs2が同じオブジェクトを参照していることを意味する

6.10 Objectクラスとclassクラス（1/5）

- Objectクラスについて
  - ObjectクラスとはJavaクラス階層の頂点に位置し
    全てのオブジェクトはObjectクラスの系列下にある
  - Objectクラスに定義された状態と動作は
    他の全てのクラスに継承される

6.10 Objectクラスと Classクラス（2/5）

- Objectクラスのメソッド
  - equals()メソッド
    構文： boolean equals(Object *obj*)
    2つの変数が同じオブジェクトを参照しているかを調べる
  - getClass()メソッド
    構文： Class getClass()
    現在のオブジェクトのClassオブジェクトを返す
  - toString()メソッド
    構文： String toString()
    現在のオブジェクトを表す文字列を返す

6.10 Objectクラスとclassクラス(3/5)

- Classクラスについて
  - 実行中のプログラム内のクラス、インターフェイスに関する情報がカプセル化されている
  - 基本データ型、void、特定の次元を持つ所定のデータ型の配列それぞれにClassオブジェクトが存在

6.10 Objectクラスとclassクラス(4/5)

- Classクラスのメソッド
  - getName()メソッド
    構文: String getName()
    データ型の名前を返す
  - get Superclass()メソッド
    構文: Class getSuperclass()
    現在のクラスのスーパークラスのClassオブジェクトを返す
  - forName()メソッド
    構文: static Class forName(String *clsName*) throws ClassNotFoundException
    名前で指定されたクラスのClassオブジェクトを返す

6.10 Objectクラスと Classクラス（5/5）

- ObjectクラスとClassクラスのメソッド使用例

```
class ClassDemo {
        public static void main(String args[] ) {
                Integer obj = new Integer( 8 );
                Class cls = obj.getClass();
                System.out.println(cls);
        }
}
```

Integerオブジェクトを作成

Classオブジェクトを取得

オブジェクトに関する
情報を表示

実行結果は
“class java.lang.Integer”と
表示される

課題

次のプログラムについて問題があれば指摘し

改善策を述べなさい。

```
class Person {
    static Person Display ;
    private String name ;
    final int age ;
    public Person( String name, int
     age ) {
        this.name = name ;
        this.age = age ;
        Display() ;
    }
    public OneYearAfter() {
        age ++ ;
        System.out.println( "1 year
     after" ) ;
        Display() ;
    }
    protected void Display() {
    System.out.println( name + " " +
     age ) ;
    }
}

class Test3 {
    public static void main( String
     args[] ) {
        Person p = new Person( "John",
     22 ) ;
        p.OneYearAfter() ;
    }
}
```