ＪａｖａプログラミングⅡ

８回目　抽象クラスとインタフェース　課題

確認○×問題

次の各文は正しいか誤っているか答えなさい。

(1)抽象クラスのオブジェクトは生成できる
(2)抽象メソッドとはメソッドの本体が未定義のメソッドである
(3)抽象メソッドをメンバーにもつクラスは抽象クラスである
(4)抽象クラスを拡張してすべての抽象メソッドをオーバーライドすれば、サブクラスのオブジェクトを生成できる
(5)インタフェースの実装は implements により行う
(6)インタフェースを実装してすべての抽象メソッドをオーバーライドすれば、そのクラスのオブジェクトを生成できる
(7)インタフェースは多重継承を実現する方法の 1 つである
(8)インタフェースは、拡張によりサブインタフェースを作ることができ、このとき拡張元をスーパーインタフェースという

■難易度★☆☆

課題 1 **パソコンの CPU、マウス、メモリなどの PC 部品を管理するクラスを PC 部品ごとに宣言しなさい。各自 2 つ以上の PC 部品を選び、次に示すインタフェース Module を実装してください。クラスのメンバーは各自にお任せしますが、3 つ以上のフィールド（変数）を宣言してください。下にハードディスクの例（この例ではフィールドは 2 つです）を示します。**

〔PC 部品を管理するインタフェースとクラス〕

```
//// Module インタフェース（※この宣言には手を加えないこと）
interface Module{
    // PC 部品のカテゴリ名を文字列として返します　"CPU", "マウス"など
    String getCategory();

    // PC 部品の情報を表示します　CPU では製造者、速度など
    void showInfor();
}

// ハードディスククラス（例）
class HardDrive implements Module{ // インタフェースの実装
    private String name;
    private int capacity;

    HardDrive(String n, int c){
        name=n;
        capacity=c;
    }
    public String getCategory(){
        return "HardDrive";
    }
    public void showInfor(){
        System.out.println(name+", "+capacity+"GB");
    }
}
```

ここにマウス Mouse やメモリ Memory などの各自 PC 部品クラスを宣言してください

■難易度★☆☆

課題2 下に示すように**main()**メソッドでインタフェース**Module**型の配列を作成しなさい。課題 1 で宣言した各自 2 つ以上の PC 部品クラスのオブジェクトを生成し、配列要素に格納しなさい。次に for 文が実行されたとき、各クラスにおいてオーバーライドをされたメソッド **getCategory()**と **showInfor()**が順次実行されることを確認しなさい。

ソースファイル名：Assignment8_2.java（main()メソッドがあるクラス名と同じにします）

```
        ここへ Module インタフェースと各 PC 部品のクラスの宣言をコピーします
class Assignment8_2{
    public static void main(String[] args){
        // 配列の作成とデータ設定
        // ※配列変数はインタフェース Module 型を用いる
        Module[] modules=new Module[3];
        modules[0]=new HardDrive("Seagate HDD",750);
        modules[1]=
        modules[2]=   ここで各自 PC 部品のクラスのオブジェクトを生成しなさい

        // 配列要素の一覧表示
        for(int i=0;i<modules.length;i++){
            System.out.println(modules[i].getCategory());
            modules[i].showInfor();
        }
    }
}
```

〔実行例〕

```
HardDrive
Seagate HDD, 750GB
  :
  :
```

■難易度★★☆

課題 3 インタフェース **Module** がもつ抽象メソッドは利用者が作成するサブクラスで必ずオーバーライドされます。この性質を利用して、インタフェース **Module** がもつ抽象メソッドを用いたコードを予め作成することができます。次はインタフェース **Module** を実装するクラスを処理する **Manager** クラスです。課題 **2** の **main()**メソッド内で **Module** 型の配列に格納されている各 **PC** 部品のクラスのオブジェクトの一覧を **Manager** クラスのクラスメソッドを用いて表示できることを確認しなさい。**Manager** クラスの宣言はコピーして下さい。

〔インタフェース Module を実装するクラスを管理する Manager クラスの宣言〕

```
// Module インタフェースを実装するクラスを処理する Manager クラス
//（※この宣言には手を加えないこと）
final class Manager{

    // 使用法　Manager.listModules(Module[]);
    // 引　数　インタフェース Module 型の配列変数
    // 戻　値　無し
    // 機　能　配列中の各 PC 部品の一覧出力
    public static void listModules(Module[] m){
        System.out.println("[パーツ一覧/パーツ数:"+m.length+"個]");
        for(int i=0;i<m.length;i++){
            System.out.println("+------------");
            System.out.println("| Parts no."+i);
            System.out.println("Category "+m[i].getCategory());
            m[i].showInfor();
            System.out.println();
        }
    }

    // 使用法　Manager.selectedModules(Module[], String);
    // 引　数　インタフェース Module 型の配列変数，カテゴリ名
    // 戻　値　無し
    // 機　能　配列中のカテゴリが一致する PC 部品の一覧
    public static void selectedModules(Module[] m, String target){
        System.out.println("[選択パーツ一覧/"+target+"]");
        for(int i=0,j=0;i<m.length;i++){
            if(m[i].getCategory()==target){
                System.out.println("+------------");
                System.out.println("| Parts no."+j++);
                System.out.println("Category "+m[i].getCategory());
                m[i].showInfor();
                System.out.println();
            }
        }
    }

    // 使用法　Manager.countModules(Module[], String);
    // 引　数　インタフェース Module 型の配列変数，カテゴリ名
    // 戻　値　カテゴリに一致する PC 部品の個数
    // 機　能　配列中のカテゴリが一致する PC 部品の数を出力
    public static int countModules(Module[] m, String target){
        int cnt=0;
        for(int i=0;i<m.length;i++)
            if(m[i].getCategory()==target)cnt++;
        return cnt;
    }
}
```

ソースファイル名：Assignment8_3.java（main()メソッドがあるクラス名と同じにします）

ここへ Module インタフェースと各 PC 部品のクラスの宣言をコピーします

ここへ Manager クラスの宣言をコピーします

```
class Assignment8_3{
    public static void main(String[] args){
        // 配列の作成とデータ設定
        // ※配列変数はインタフェース Module 型を用いる
        Module[] modules=new Module[3];
        modules[0]=new HardDrive("Seagate HDD",750);
        modules[1]=
        modules[2]=

        // ＰＣ部品一覧
        Manager.listModules(modules);

        // ハードディスクのみの一覧と個数
        Manager.selectedModules(modules,modules[0].getCategory());
        int cnt=Manager.countModules(modules,modules[0].getCategory());
        System.out.println(" 個数 "+cnt);
    }
}
```

ここで各自 PC 部品のクラスのオブジェクトを生成しなさい

〔実行例〕

```
 [パーツ一覧/パーツ数:3 個]
+------------
| Parts no.0
Category HardDrive
Seagate HDD, 750GB
   :

[選択パーツ一覧/HardDrive]
+------------
| Parts no.0
Category HardDrive
Seagate HDD, 750GB
   :
```

■難易度★★★

課題 4 演算子を管理する Operation インタフェースは次の抽象メソッドを持ちます。

1．演算子のオペランドの数を返す抽象メソッド
2．演算子のオペランドを設定する抽象メソッド
3．設定したオペランドを用いた式（質問）を生成し、生成した式を返す抽象メソッド
4．設定したオペランドを用いた式を計算し、結果を返す抽象メソッド

〔Operation インタフェースの宣言〕

```
// Operation インタフェース （※この宣言には手を加えないこと）
interface Operation{
   int getNumOfOperands();            // オペランドの数を返します
   void setOperands(double[] values); // オペランドを設定します
   String getExpression();            // 設定したオペランドで式を生成します
   double getAnswer();                // 設定したオペランドで式の結果を返します
}
```

(1) Operation インタフェースを実装した各自オリジナルの演算子クラスを宣言しなさい。
以下のコードに距離演算子クラスと加算演算子クラスの例を示します。

〔Operation インタフェースを実装したクラス群〕

```
// 距離演算子クラス（演算子インタフェース実装例 1）
class Distance implements Operation{
    private double x1, y1, x2, y2;

    public int getNumOfOperands(){
        return(4);
    }
    public void setOperands(double[] values){
        x1=values[0]; y1=values[1];
        x2=values[2]; y2=values[3];
    }
    public String getExpression(){
        return("座標("+x1+","+y1+")と座標("+x2+","+y2+")間の距離?");
    }
    public double getAnswer(){
        return(Math.sqrt(Math.pow(x1-x2,2)+Math.pow(y1-y2,2)));
    }
}

// 加算演算子クラス（演算子インタフェース実装例 2）
class Addition implements Operation{
    private double n1, n2;

    public int getNumOfOperands(){
        return(2);
    }
    public void setOperands(double[] values){
        n1=values[0]; n2=values[1];
    }
    public String getExpression(){
        return(n1+"+"+n2+"?");
    }
    public double getAnswer(){
        return(n1+n2);
    }
}
```

ここに各自の○○演算子クラスを宣言してください
例えば　・引算クラスでは、2 つの値がオペランドでその差が演算結果です
　　・円面積クラスでは、1 つの値（半径）がオペランドで πr^2 が演算結果です

(2) クラス PrintQuestion は Operation インタフェースを実装するクラスのオブジェクトを用いて問題と解答を表示する機能を持ちます。以下の実行例に示すように、このクラスのクラスメソッド void prtWithRand(Operation op)を用いて各自の演算子クラスの問題を画面に出力しなさい。

ソースファイル名：Assignment8_4.java（main()メソッドがあるクラス名と同じにします）

```
        ここへ Operation インタフェースと各演算子クラスの宣言をコピーします

////////////////////////////////////////////////
// 問題生成クラス （※この宣言には手を加えないこと）
final class PrintQuestion{
    // 演算子インタフェースを実装したクラスのオブジェクト用い、
    // 乱数でオペランドを与えながら問題を作成します
    public static void prtWithRand(Operation op){
        double[] values=new double[op.getNumOfOperands()];
        for(int i=0;i<values.length;i++)
            values[i]=(int)(Math.random()*10);
        op.setOperands(values);
        System.out.println("Q."+op.getExpression());
        System.out.println("A."+op.getAnswer());
    }
    // 演算子インタフェースを実装したクラスのオブジェクト用い、
    // 与えられたオペランドで問題を作成します
    public static void prtWithValues(Operation op, double[] values){
        op.setOperands(values);
        System.out.println("Q."+op.getExpression());
        System.out.println("A."+op.getAnswer());
    }
}

class Assignment8_4{
    public static void main(String[] args){
        Distance dis=new Distance();
        Addition add=new Addition();
          ここで各自の演算子クラスのオブジェクトを生成してください

        PrintQuestion.prtWithRand(dis);
        PrintQuestion.prtWithRand(add);
          ここで各自の演算子の問題を生成してください

    }
}
```

〔実行例〕

```
Q.座標(6.0,1.0)と座標(6.0,2.0)間の距離?
A.1.0
Q.6.0+9.0?
A.15.0
  :
```