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。

⑴　抽象クラスのオブジェクトは生成できる
⑵　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
⑶　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
⑷　抽象クラスを継承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
⑸　インタフェースの実装は implements により行う
⑹　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
⑺　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
⑻　インタフェースは、継承によりサブインタフェースを作ることができ、このとき継承元をスーパーインタフェースという

課題１　パソコンのＣＰＵ、マウス、メモリなどのモジュールを管理するクラスをモジュールごとに宣言しなさい。各自２つ以上のモジュールを選び、次に示すインタフェース Module を実装して宣言すること。下にハードディスクの一例を示す。

```
interface Module
{
    // モジュールカテゴリ　カテゴリ名を文字列として返す“ＣＰＵ”,“マウス”など
    String getCategory();

    // モジュールの情報　ＣＰＵでは製造者、速度などを表示する
    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");
    }
}
```

課題２　メインメソッドでインタフェース Module 型の配列を下のように作成しなさい。課題１で宣言した各自の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。次に下のコードのように for 文を実行して、各オブジェクト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。

```
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]= ・・・・・// 各自宣言したクラスのオブジェクトを生成
		modules[2]= ・・・・・// 各自宣言したクラスのオブジェクトを生成

		// 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();
		}
	}
}
```

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る。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る。次はインタフェース Module を実装するクラスのオブジェクトを処理するクラス Manager である。

```
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　配列中の各モジュールの一覧出力
    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モジュールの一覧
    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モジュールの個数
    //// 機　能　配列中のカテゴリが一致するモジュールの数を出力
    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;
    }
}
```

このクラス宣言を各自のファイルにコピーしなさい。メインメソッド内でこのクラスメソッドを用いて Module 型の配列に格納されているオブジェクトの一覧を表示しなさい。

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つ。

１、　演算子のオペランドの数の取得
２、　演算子のオペランドの設定
３、　設定したオペランドを用いた式の生成
４、　設定したオペランドを用いた式の計算

```
////////////////////////////////////////////////////////
// 演算子インタフェース （※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）この Operation インタフェースを実装した各自オリジナルの演算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。

```
////////////////////////////////////////////////////
// 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);
    }
}
```

（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)を用いて各自オリジナルの演算クラスの問題を画面に出力しなさい。

```
//////////////////////////////////////////////////
// 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());
        }
}
```

《Main メソッド例》

```
class Assignment8_4{
        public static void main(String[] args){
                Distance dis=new Distance();
                Addition add=new Addition();
                // <--ここでオリジナル演算クラスのオブジェクトの生成

                PrintQuestion.prtWithRand(dis);
                PrintQuestion.prtWithRand(add);
                // <--ここでオリジナル演算の問題の生成
        }
}
```

《実行例》

```
>java Assignment8_4
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
 -- Press any key to exit (Input "c" to continue) --
```