

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
- (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、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。