Chapter 08 接口

Key Point：

- ●接口的基本语法
- ●接口的作用
- ●解耦合

问题：

1. 代码改错：

```
interface IA{
    void m1();
    int a = 100;
}
class MyClass implements IA{
    void m1(){}
}
public class TestInterface{
    public static void main(String args[]){
        IA ia = new MyClass();
        ia.m1();
        System.out.println(IA.a);
    }
}
```

2. 代码填空：

```
interface IA{
    void m1();
    void m2();
}
_____ class MyClassA implements IA{
    public void m1(){}
    }
class MyClassB extends MyClassA{
    _____ {}
}
```

3. 有如下代码：

```
interface IA{
    void ma();
}
interface IB extends IA{
    void mb();
}
interface IC{
    void mc();
}
interface ID extends IB, IC{
    void md();
```

}

I. 如果有一个类 ClassE 实现 ID 接口，如果不希望 ClassE 是抽象的，则需要实现哪些方法？

II. 把下面的代码补充完整

```
public class TestClassE{
    public static void main(String args[]){
        IC ic = new ClassE();
        //调用 ma 方法
        _____
        //调用 mb 方法
        _____
        //调用 mc 方法
        _____
        //调用 md 方法
        _____
    }
}
```

III. 写出下面代码的输出结果

```
public class TestClassE{
    public static void main(String args[]){
        IC ic = new ClassE();
        System.out.println(ic instanceof IA);
        System.out.println(ic instanceof IB);
        System.out.println(ic instanceof IC);
```

```
            System.out.println(ic instanceof ID);
            System.out.println(ic instanceof ClassE);
        }
    }
```

4. 有如下代码：

```
    interface IA{
        void ma();
    }
    interface IB{
        void mb();
    }
    class MySuper implements IA{
        public void ma(){}
    }
    class MySub extends MySuper implements IB{
        public void mb(){}
    }
    public class TestMain{
        public static void main(String args[]){
            MySuper ms = new MySub();
            System.out.println(ms instanceof IA);
            System.out.println(ms instanceof IB);
            System.out.println(ms instanceof MySuper);
```

```
        System.out.println(ms instanceof MySub);
    }
}
```

问：该程序输出结果是什么？

5. 关于接口和抽象类，下列说法正确的是：

A． 抽象类可以有构造方法，接口没有构造方法

B． 抽象类可以有属性，接口没有属性

C． 抽象类可以有非抽象方法，接口中都是抽象方法

D． 抽象类和接口都不能创建对象

E． 一个类最多可以继承一个抽象类，但是可以实现多个接口

6. 写出下面代码的输出结果

```
interface Light{
    void shine();
}
class RedLight implements Light{
    public void shine(){
        System.out.println(“Red Light shine in Red”);
    }
}
class YellowLight implements Light{
    public void shine(){
        System.out.println(“Yellow Light shine in Yellow”);
```

```
    }
}
class GreenLight implements Light{
    public void shine(){
        System.out.println( “Green Light shine in Green” );
    }
}
class Lamp{
    private Light light;
    public void setLight(Light light){
        this.light = light;
    }
    public void on(){
        light.shine();
    }
}
public class TestLamp{
    public static void main(String args[]){
        Light[] ls = new Light[3];
        ls[0] = new RedLight();
        ls[1] = new YellowLight();
        ls[2] = new GreenLight();
        Lamp lamp = new Lamp();
        for (int i = 0; i<ls.length; i++){
```

```
                lamp.setLight(ls[i]);
                lamp.on();
            }
        }
    }
```

7. 写出下面代码执行的结果

```
interface JavaTeacher{
    void teach();
}
class TeacherA implements JavaTeacher{
    public void teach(){
        System.out.println(“TeacherA teach Java”);
    }
}
class TeacherB implements JavaTeacher{
    public void teach(){
        System.out.println(“TeacherB teach Java”);
    }
}
class School{
    public static JavaTeacher getTeacher(int i){
        if (i == 0) return new TeacherA();
            else return new TeacherB();
```

```
            }
    }
    public class TestSchool{
        public static void main(String args[]){
            JavaTeacher jt = School.getTeacher(0);
            jt.teach();
            jt = School.getTeacher(10);
            jt.teach();
        }
    }
```

8. 代码填空

```
    abstract class Animal{
        public abstract void eat();
    }
    interface Pet{
        void play();
    }
    class Dog extends Animal implements Pet{
        public void eat(){
            System.out.println(“Dog eat Bones”);
        }
        public void play(){
            System.out.println(“Play with Dog”);
```

```
    }
}
class Cat extends Animal implements Pet{
    public void eat(){
        System.out.println( “Cat eat fish” );
    }
    public void play(){
        System.out.println( “Play with Cat” );
    }
}
class Wolf extends Animal{
    public void eat(){
        System.out.println( “Wolf eat meat” );
    }
}
public class TestMain{
    public static void main(String args[]){
        Animal as[] = new Animal[3];
        as[0] = new Dog();
        as[1] = new Cat();
        as[2] = new Wolf();
        //调用 as 数组中所有动物的 eat 方法
        //1
        //调用 as 数组中所有宠物的 play 方法
```

```
            //2
        }
    }
```

//1 处应该填入的代码为_____。

//2 处应该填入的代码为_____。

9.在原有的 Chap6 中的 17 题基础之上修改代码

公司给 SalariedEmployee 每月另外发放 2000 元加班费，给 BasePlusSalesEmployee 发放 1000 元加班费改写原有代码，加入以上的逻辑。并写一个方法，打印出本月公司总共发放了多少加班费。

10. 有下列代码：

```
interface ServiceInterface{
    void doService1();
    void doService2();
    void doService3();
}
abstract class AbstractService implements ServiceInterface{
    public void doService1(){}
    public void doService2(){}
    public void doService3(){}
}
```

需要一个实现 ServiceInterface 接口的类 MyService。

I. 第一种方式可以让 MyService 实现 ServiceInterface 接口，即：

```
class MyService implements ServiceInterface
```

II. 第二种方式可以让 MyService 继承 AbstractService 类，即：

class MyService extends AbstractService

请问：这两种方式有什么区别？AbstractService 类有什么作用？

11. 验证歌德巴赫猜想

输入一个大于 6 的偶数，请输出这个偶数能被分解为哪两个质数的和。

如 10=3+7 12=5+7

要求：两个人一组合作完成。一个人负责把一个整数 n 拆分成两个整数的和，另一个人负责写一个函数，判断某一个整数 a 是否是质数 。