Day13 - 接口

百知教育 Baizhi Education

应用知识点：

- **接口的基本语法**
- **接口的继承性**
- **接口的作用**

习题：

1. 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单独创建对象

2. 下列哪一种叙述是正确的（ ）

   A. 一个 Java 类只能实现一个接口

   B. 一个 Java 类不能同时继承一个类和实现一个接口

   C. 一个 Java 类可以实现多个接口

   D. 一个 Java 类只能直接继承一个类

3. 仔细阅读以下代码，有几处错误代码？并修改正确。

```
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);
    }
}
```

4. 仔细阅读以下代码，根据语法将代码补全。

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

5. 仔细阅读以下代码，根据要求完成程序。

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

(1) 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？

(2) 将以下代码补充完整。

```
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
        _____________
    }
}
```

(3) 写出以下程序输出的结果。

```
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);
    }
}
```

6. 仔细阅读以下代码，写出程序输出的结果。

```
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);
    }
}
```

7. 仔细阅读以下代码，写出程序运行输出结果。

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

8. 仔细阅读以下代码，写出程序输出的结果。

```
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();
    }
}
```

9. 仔细阅读以下代码：

```
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。

(1) 第一种方式可以让 MyService 实现 ServiceInterface 接口，即：

class MyService implements ServiceInterface

(2) 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？

10. 仔细阅读以下代码，完成//1、//2、//3、//4 处程序。

```
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[]) {
        //1.定义一个Animal类型as数组，存储多个不同子类型对象
        //2.调用as数组中所有动物的eat 方法
        //3.统计数组中宠物的个数
        //4.调用as数组中所有宠物的play方法
    }
}
```

11. 编程：定义一个接口 MathTool，接口中有三个抽象方法如下：

    (1) “long fact(int m);”方法的功能为：求参数的阶乘

    (2) “long intPower(int m , int n)”方法的功能为：求 m 的 n 次方

    (3) “boolean findFactor(int m,int n)”方法的功能为：判断参数的和是否大于 100

    定义类实现接口，编写应用程序，调用接口中的 3 个方法，并将调用方法的结果输出。

12. 编程：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 。