### JAVA 第一阶段—DAY10-JAVA作业答案

1. 简答题：简述什么抽象类以及注意事项。

**答案：**

抽象类：包含有抽象方法的类就是抽象类，使用 abstract 关键字修饰方法，该方法就成了抽象方法，抽象方法只包含一个方法名，而没有方法体。

注意事项：

- 抽象类不能创建对象。

- 抽象类中，可以有构造方法。

- 抽象类中，不一定包含抽象方法，但是有抽象方法的类必定是抽象类。

- 抽象类的子类，必须重写抽象父类中所有的抽象方法，否则，编译无法通过而报错。除非该子类也是抽象类。

1. 简答题：简述JDK8中接口可以包括有哪些内容。

**答案：**

常量、抽象方法、静态方法、默认方法。

1. 简答题：简述Java中抽象方法、静态方法和默认方法的注意事项。

**答案：**

1. **抽象方法**

接口中抽象方法的定义与抽象类中方法的定义及使用是一样的；

接口中，定义抽象方法时，abstract关键字可以省略，JVM会自动添加。

1. **静态方法**

- 静态方法使用 static 关键字修饰，不可省略，供接口直接调用；

- 静态方法不能被继承，也不能被重写；

- 静态方法只能通过接口名直接调用。

1. **默认方法**

- 默认方法使用 default 关键字修饰，不可省略，供子类调用；

- 默认方法可以被子类继承、同时也可以被子类实现，被子类实现时必须去除default关键字。

1. 简答题：简述Java中接口多实现的注意事项。

**答案：**

- 子类进行多接口实现时，如果接口常量没有重名，可以按照之前的方式进行使用；如果接口常量出现重名，就必须通过接口名调用常量。

- 子类进行多接口实现时，接口中有多个抽象方法时，实现类必须重写所有抽象方法。如果抽象方法有重名的，只需要重写一次。

- 子类进行多接口实现时，接口中存在同名的静态方法并不会冲突，原因是只能通过各自接口名访问静态方法。

- 子类进行多接口实现时，接口中有多个默认方法时，实现类都可以选择继承使用。如果默认方法有重名的，必须重写一次。

- 当一个类既继承一个父类，又实现若干个接口时，父类中的成员方法与接口中的默认方法重名，子类就近选择执行父类的成员方法。

1. 简答题：阅读以下描述，使用代码完成案例。

**答案：**

//1、定义衣服类

public class Clothes {

// a)成员变量:

// i.颜色(color)

private String color;

// ii.品牌(brand)

private String brand;

// b)提供 无参,带参构造和setXxx和getXxx方法

public Clothes() {

super();

}

public Clothes(String color, String brand) {

super();

this.color = color;

this.brand = brand;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

public String getBrand() {

return brand;

}

public void setBrand(String brand) {

this.brand = brand;

}

}

// 2、定义抽象人类

public abstract class Person {

//a)成员变量

// i.姓名 name

private String name;

// ii.年龄 age

private int age;

// iii.身高 height

private double height;

//b)成员方法

// i.抽象方法 void say(String content)

public abstract void say(String content);

//c)要求

// i.提供无参和带参构造

public Person() {

super();

}

public Person(String name, int age, double height) {

super();

this.name = name;

this.age = age;

this.height = height;

}

// ii.提供setXxx和getXxx方法

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

public double getHeight() {

return height;

}

public void setHeight(double height) {

this.height = height;

}

}

// 3、定义女生类

public class Girl extends Person {

//a)成员变量

// i.脸型(String face)

private String face;

//b)成员方法:

// i.重写父类的 void say(String content) 方法

// 1.输出格式: 欧阳青青微笑着用于甜美的声音对慕容紫英说:有你在我就安心.

public void say(String countent){

System.out.println(getName()+"微笑着用于甜美的声音对慕容紫英说:"+countent);

}

// ii.特有方法: 洗衣服 wash(Clothes c)

// 输出格式: 欧阳青青在洗一件白色的李宁牌子的衣服

public void wash(Clothes c){

System.out.println(getName()+"在洗一件"+c.getColor()+"的"+c.getBrand()+"牌子的衣服");

}

//c)要求:

// i.提供无参和带参构造

// ii.提供setXxx和getXxx方法

public Girl() {

super();

}

public Girl(String name, int age, double height,String face) {

super(name, age, height);

this.face = face;

}

}

// 4、定义男生类

public class Boy extends Person {

//a)成员变量

// i.女朋友 (girlFirend)

private Girl girlFriend;

//b)成员方法:

// i.重写父类的 void say(String content) 方法

// 输出格式:容紫英用带有阳刚之气的声音对欧阳青青说:我会守护你一生一世

public void say(String content){

System.out.println(getName()+"带有阳刚之气的声音对"+girlFriend.getName()+"说:"+content);

}

// ii.特有方法: 散步 walking()

// 输出格式: 慕容紫英和欧阳青青在河边的林荫小道上手牵着手散步

public void walking(){

System.out.println(getName()+"和"+girlFriend.getName()+"在河边的林荫小道上手牵着手散步");

}

//c)要求:

// i.提供无参和带参构造

public Boy() {

super();

}

public Boy(String name, int age, double height,Girl girlFriend) {

super(name, age, height);

this.girlFriend = girlFriend;

}

// iii.提供setXxx和getXxx方法

public Girl getGirlFriend() {

return girlFriend;

}

public void setGirlFriend(Girl girlFriend) {

this.girlFriend = girlFriend;

}

}

// 5、定义测试类

public class Test {

public static void main(String[] args) {

// i.创建衣服对象 c,品牌初始化为李宁,颜色初始化为白色

Clothes c = new Clothes("白色", "李宁");

// ii.创建女孩对象 girl,姓名赋值为欧阳青青,年龄赋值为18,身高赋值为1.6,脸型赋值为瓜子脸

Girl girl = new Girl("欧阳青青", 18, 1.6, "瓜子脸");

// iii.创建男孩对象 boy,姓名赋值为慕容紫英,年龄赋值以为20,身高赋值1.75,女朋友赋值为girl

Boy boy = new Boy("慕容紫英", 20, 1.75, girl);

// iv.调用对象girl的洗衣服方法

girl.wash(c);

// v.调用对象boy的散步方法

boy.walking();

// vi.调用对象boy的说话方法,传入:我会守护你一生一世

boy.say("我会守护你一生一世.");

// vii.调用对象 girl 的说话方法,传入: 有你在我就安心.

girl.say("有你在我就安心.");

}

}

1. 编程题：自定义一个类型数组，使用两种排序方式进行排序。

**答案：**

// 学生类

class Student {

private String name;

private int age;

public Student() {

}

public Student(String name, int age) {

this.name = name;

this.age = age;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

@Override

public String toString() {

return "Student [name=" + name + ", age=" + age + "]";

}

}

// 测试类

public class CompareTest {

public static void main(String[] args) {

// 初始化学生类数组

Student[] str=new Student[5];

str[0]=new Student("tom",22);

str[1]=new Student("kitty",18);

str[2]=new Student("Jacklove",23);

str[3]=new Student("uzi",23);

str[4]=new Student("tom",17);

// 打印原生数组

System.out.println(Arrays.toString(str));

// 工具类默认排序

Arrays.sort(str);

// 排序后再次打印

System.out.println(Arrays.toString(str));

}

}