抽象类+接口+内部类作业题

一、 选择题

1. Person 类和 Test 类的代码如下所示，则代码中的错误语句是（ C ）。（选择一项）

```
public class Person {
    public String name;
    public Person(String name) {
        this.name = name;
    }
}
public class Test {
    public static void main(String[] args) {
        final Person person = new Person("欧欧");
        person.name = "美美";
        person = new Person("亚亚");
    }
}
```

A. final Person person = new Person("欧欧");

B. person.name = "美美";

C. person = new Person("亚亚");

D. 没有错误

2. 如下 Java 代码运行的结果是（ B ）。（选择一项）

```
public class Test {
    final int age;
    public Test(){
        age=10;
    }
    public static void main(String[ ] args) {
        System.out.println(new Test().age);
    }
}
```

A 输出：0

B. 输出：10

C. 输出：null

D. 运行时出现异常

3. 以下代码中错误的语句是（ B ）。（选择一项）

```
public class Something{
    public static void main(String[] args){
```

```
        final Other o=new Other();
        new Something().addOne(o);//1
    }
    public void addOne( Other o){
        o.i++;//2
        o =   new Other();//3
    }
}
class Other{
    public int i;
}
```

A 1

B. 2

C. 3

D. 没有错误

4. **下列选项中，关于Java的抽象类和抽象方法说法正确的是（AC）。（选择二项）**

A 抽象类中可以含有 0 个或多个抽象方法

B. 抽象类中不可以有构造方法

C. 一个类中若有抽象方法，则这个类必为抽象类

D. 子类必须重写父类所有的抽象方法

5. **在 Java 中关于 abstract 关键字，以下说法正确的是（AB）。（选择两项）**

A abstract 类中可以没有抽象方法

B. abstract 类的子类也可以是抽象类

C. abstract 方法可以有方法体

D. abstract 类可以创建对象

6. **以下选项可替换题目中//add code here 而不产生编译错误的是（BD）。（选择二项）**

```
public abstract class MyClass{
    public int constInt=5;
    //add code here
    public void method( ){
    }
}
```

A public abstract void method(int a);

B. constInt =constInt+5;

C. public int method( );

D. public abstract void anotherMethod( );

7. **在 Java 接口中，下列选项中属于有效的方法声明是（ AB ）。（选择二项）**

A public void aMethod( );

B. final void aMethod( );

C. void aMethod(){ }

D. private void aMethod( );

8. **以下选项中关于匿名内部类的说法正确的是（ AB)。（选择二项）**

A. 匿名内部类可以实现多个接口，或者继承一个父类

B. 匿名内部类不能是抽象类，必须实现它的抽象父类或者接口里包含的所有抽象方法

C. 匿名内部类没有类名，所以匿名内部类不能定义构造方法

D. 匿名内部类可以直接访问外部类的所有局部变量

9. **接口定义正确的说法是（ AD ）。（选择二项）**

A. 接口中只能定义常量和抽象方法

B. 接口中可以定义普通方法和普通变量

C. 接口可以被 new，抽象方法不行

D. 接口不可以被继承，只能被实现，也可以被多个类实现

二、 判断题（共 20 个题目，总计 10 分）

1. 声明为 final 的类不能是超类。（ T ）
2. 使用 final 修饰的变量将变成常量，其中不能再被改变；使用 final 修饰的方法将无法被子类重载；使用 final 修饰的类将无法再被其他类继承。（ F ）
3. 抽象方法不能含有方法体，并且必须定义在抽象类中。（ T ）
4. 抽象类是指在 class 前加使用 abstract 关键字修饰，且可以存在抽象方法和普通方法的类。（ T ）
5. 接口中只有常量，没有变量；只有抽象方法，并且全部都是 public 方法。（ T ）
6. 抽象类和接口都不能实例化，都没有构造方法。（ F ）
7. 接口并不要求实现类和接口在概念本质上一致的，仅仅是实现了接口定义的约定或者能力而已。接口定义了“做什么”，而实现类负责完成“怎么做”，体现了功能（规范）和实现分离的原则。（ F ）
8. 内部类作为外部类成员，权限修饰符和其他成员一样，可声明为 private、默认、protected 或 public。(T)
9. 匿名内部类适合创建那种只需要使用一次的类，它可以实现一个或多个接口，或者继承一个父类。（ T ）
10. 对于物理连接，比如数据库连接、输入流输出流、Socket 连接等，垃圾回收机制无能为力，必须手动关闭才可以。(F)
11. 垃圾回收机制回收任何对象之前，总会先调用它 gc( )方法，该方法是 Object 类提供的方法。不要主动调用某个对象的该方法，应该交给垃圾回收机制调用。（ T ）
12. final 修饰的方法不能被重写，但可以被重载（ F ）
13. 抽象方法的类必须是抽象类，同样抽象类也必须包含抽象方法（ F ）
14. 抽象类可以定义普通方法，但不能定义构造方法（ T ）
15. 抽象类不能被 new，抽象类的构造方法也不能被调用（ T ）

三、 简答题

1. final 修饰变量、方法、类分别表示什么？
2. final 和 abstract 关键字的作用。
3. 接口和抽象类的联系和区别。
4. java 中有多继承吗?如果没有，java 为什么取消了多继承？

C++中有多继承吗？单继承相比多继承优势在什么地方?

5. 内部类的类型及其特点。
6. 介绍 Java 垃圾回收机制。

四、 编码题

1. 编写程序描述兔子和青蛙

需求说明：使用面向对象的思想，设计自定义类描述兔子和青蛙。

实现思路及关键代码

1) 分析兔子和青蛙的共性
2) 根据共性，定义抽象的动物类

属性：名字、颜色、类别（哺乳类、非哺乳类）

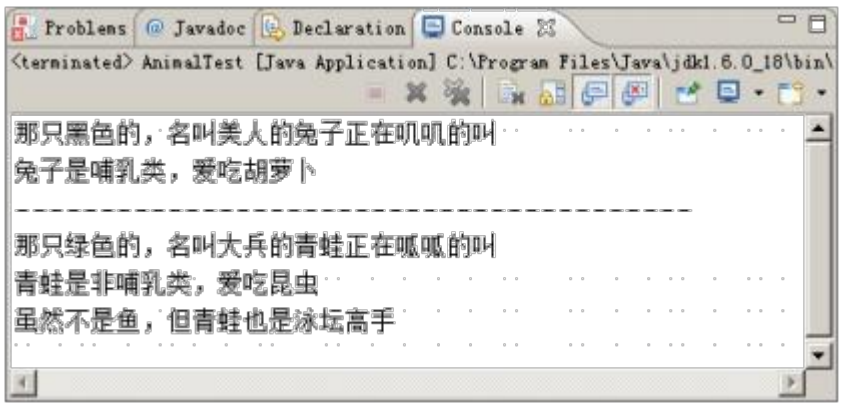
方法：吃饭，发出叫声

3) 根据青蛙会游泳 抽象游泳的接口

方法：游泳

4) 定义兔子继承动物类，青蛙继承动物同时实现游泳接口

程序运行结果如图所示。



2. 编写程序描述影视歌三栖艺人

需求说明：请使用面向对象的思想，设计自定义类，描述影视歌三栖艺人。

实现思路及关键代码

1) 分析影视歌三栖艺人的特性
   a) 可以演电影
   b) 可以演电视剧
   c) 可以唱歌
2) 定义多个接口描述特性
   a) 演电影的接口-----方法：演电影
   b) 演电视剧的接口-----方法：演电视剧
   c) 唱歌的接口-----方法：唱歌
3) 定义艺人类实现多个接口

程序运行结果如图 2 所示。

1　final 修饰变量、方法、类分别表示什么?

final修饰类是不能被继承

fianl修饰方法不能在子类中被覆盖

final修饰变量，称为常量，初始化以后不能改变值。

2　final和abstract关键字的作用。

Final修饰完的都变为常量或不可再改变的方法，被abstract修饰的类变为抽象类，被修饰的方法变为抽象方法，被继承必须重写抽象方法。

3　接口和抽象类的联系和区别。

抽象类中可以没有抽象方法，接口中如果有方法一定是抽象方法。都是抽象的

4　java 中有多继承吗?如果没有，java 为什么取消了多继承?

没有，因为Java有接口

5　内部类的类型及其特点。

成员内部类：成员内部类，就是作为外部类的成员，可以直接使用外部类的所有成员和方法，即使是private的。同时外部类要访问内部类的所有成员变量/方法，则需要通过内部类的对象来获取。实例化方式：new Out().new In();

局部内部类：是指内部类定义在方法和作用域内。

静态内部类：其实它和成员内部类很相似，只是只能访问外部类的静态成员变量，具有局限性。实例化方式：new Out.In();其实你会发现就是把内部类看成一个属性来对待。

匿名内部类：匿名内部类不能加访问修饰符。其实我们最常见的执行一个线程就是用的内部类的方式。

6　介绍 Java 垃圾回收机制。

当一个对象的地址没有变量去引用时，该对象就会成为垃圾对象，垃圾回收器在空闲的时候会对其进行内存清理回收

1

```java
public abstract class Animal {
    protected String name;
    3 个用法
    protected String color;

    2 个用法  2 个实现  新 *
    public abstract void eat();
    2 个用法  2 个实现  新 *
    public abstract void bark();

    2 个用法  新 *
    public Animal(String name, String color) {
        this.name = name;
        this.color = color;
    }
}
```

```java
public interface Swim {
    1 个用法  1 个实现  新 *
    void Swim();
}
```

```java
public class Frog extends Animal implements Swim{
    1 个用法
    private final String classification = "非哺乳类";

    1 个用法  新 *
    public Frog(String name, String color) {
        super(name, color);
    }

    2 个用法  新 *
    @Override
    public void eat() { System.out.println("青蛙是".concat(classification).concat( str: ", 爱吃昆虫")); }

    2 个用法  新 *
    @Override
    public void bark() {
        System.out.println("那只".concat(this.color).concat( str: "的, 名叫").concat(this.name).concat( str: "的请外正在呱呱
    }

    1 个用法  新 *
    @Override
    public void Swim() { System.out.println("虽然不是鱼, 但青蛙也是泳坛高手"); }
}
```

```java
public class Rabbit extends Animal{
    1 个用法
    private final String classification = "哺乳类";

    1 个用法  新 *
    public Rabbit(String name, String color) {
        super(name, color);
    }

    2 个用法  新 *
    @Override
    public void eat() {
        System.out.println("兔子是".concat(this.classification).concat( str: ", 爱吃胡萝卜"));

    }

    2 个用法  新 *
    @Override
    public void bark() {
        System.out.println("那只".concat(this.color).concat( str: "的, 名叫").concat(this.name).concat( str: "的兔子正在叽
    }
}
```

```java
public class Test {
    新 *
    public static void main(String[] args) {

        Animal a1 = new Rabbit( name: "美人",  color: "黑色");
        a1.bark();
        a1.eat();
        Frog a2 = new Frog( name: "大兵",  color: "黑色");
        a2.bark();
        a2.eat();
        a2.Swim();

    }
}
```

2

```java
public class Idol implements IsInTheMovie, InTheTVPlay, Sing{
    2个用法
    private String name;

    1个用法  新 *
    public Idol(String name) {
        this.name = name;
    }

    1个用法  新 *
    @Override
    public void inTheTvPlay() { System.out.println("我能演电视剧"); }

    1个用法  新 *
    @Override
    public void isInTheMovie() { System.out.println("我能演电影"); }

    1个用法  新 *
    @Override
    public void sing() {
        System.out.println("我会唱歌");

    }
    新 *
    public void show(){
        System.out.println("大家好! 我是" + this.name);
        inTheTvPlay();
        isInTheMovie();
        sing();
    }
}
```

```java
public interface InTheTVPlay {
    1个用法  1个实现  新 *
    void inTheTvPlay();
}
```

```java
public interface Sing {
    1个用法  1个实现  新 *
    void sing();
}
```

```java
public interface IsInTheMovie {
    1个用法  1个实现  新 *
    void isInTheMovie();
}
```

```java
public class Test {
    新 *
    public static void main(String[] args) {
        Idol idol = new Idol( name: "马素素");
        idol.show();
    }
}
```

3

```java
public class Person implements Comparable<Person>{
    public String name;
    11 个用法
    public Integer age;

    10 个用法  新 *
    public Person(String name, Integer age) {
        this.name = name;
        this.age = age;
    }


    1个用法  新 *
    public void sort(Object[] obj) {
        Person[] persons=(Person[])obj;
        for (int i = 0; i < persons.length; i++) {
            for (int j = 0; j < persons.length - 1; j++) {
                Person[] temp = new Person[1];
                if (persons[j].age > persons[j + 1].age) {
                    temp[0] = persons[j];
                    persons[j] = persons[j + 1];
                    persons[j + 1] = temp[0];
                }
            }
        }
    }
```

```java
    新 *
    @Override
    public int compareTo(Person p) {
        if(Objects.equals(this.age, p.age)){
            return 0;
        }else if (this.age > p.age) {
            return 1;
        }else {
            return -1;
        }
    }
}
```

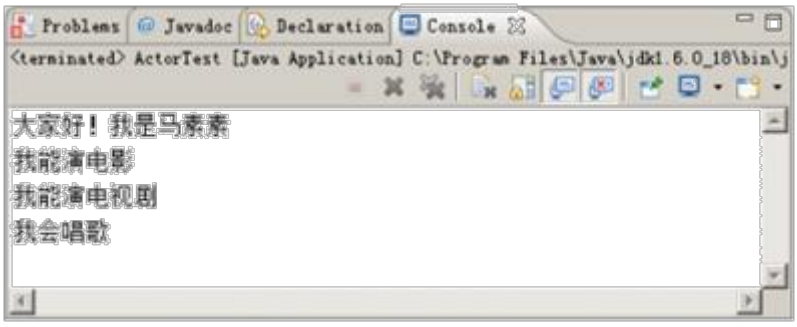
```java
public class SortReference{
    新 *
    public static void main(String[] args) {

        Person p1 = new Person( name: "A",  age: 10);
        Person p2 = new Person( name: "B",  age: 8);
        Person p3 = new Person( name: "C",  age: 7);
        Person p4 = new Person( name: "D",  age: 11);
        Person p5 = new Person( name: "E",  age: 12);
        Person p6 = new Person( name: "F",  age: 21);
        Person p7 = new Person( name: "G",  age: 18);
        Person p8 = new Person( name: "H",  age: 19);
        Person p9 = new Person( name: "I",  age: 221);
        Person p10 = new Person( name: "J",  age: 110);
        Person[] p = {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10};
        Person[] pp = {p1, p2, p3, p4, p5, p6, p7, p8, p9, p10};

        for (Person person : p) {
            System.out.print("姓名" + person.name + ", 年龄" + person.age + "\t");
        }
        System.out.println("================");

        Arrays.sort(p);
        for (Person person : p) {
            System.out.print("姓名" + person.name + ", 年龄" + person.age+ "\t");
        }
        System.out.println("================");

        for (Person person : pp) {
            System.out.print("姓名" + person.name + ", 年龄" + person.age+ "\t");
        }
        System.out.println("================");
```



3. 写一个方法对任意引用数据类型数组进行排序。具体要求如下:
   1) 方法声明为public void sortArr(Object arr[]){ }
   2) 方法中首先输出排序前数组内容，然后进行排序，最后输出排序后数组内容。
   3) 可以是冒泡排序或其他算法实现，不直接调用Java提供的方法实现排序。
      思路：任意类实现Comparable接口来实现该引用数据类型的元素排序，在sort()方法中将Object强转成Comparable实现两个对象的比较。

五、 可选题

1. 实现不同符合 PCI 规范的适配器
   需求说明：PCI 是一种规范，所有实现了该规范的适配器，必如显卡、声卡、网卡都可以安装到 PCI 插槽上并工作。模拟实现该功能。

   实现思路及关键代码
   1) 定义 PCI 接口，具有传送数据 send()方法
   2) 定义显卡 VideaCard 类，实现该接口
   3) 定义声卡 AudioCard 类，实现 PCI 接口
   4) 定义网卡 NetCard 类，实现 PCI 接口
   5) 定义测试类，让显卡、声卡、网卡发送数据

2. 实现不同引用类型对象的大小比较
   需求说明：学生类，新闻类，商品类虽然是完全不同的类，但是都具有比较的能力，比如可以比较两个学生的大小，但需要指定比较的依据是学号、姓名还是成绩等。
   实现思路及关键代码：
   将比较的能力定义为接口，让学生、新闻、商品类都实现该接口。
   1) 定义接口 Comparable，其中包含唯一的方法 int compareTo(Object obj)；返回值>0，

表示大于，返回值=0，表示等于，返回值<0，表示小于。

2) 定义学生类，包括学号、姓名、年龄和分数，实现 Comparable 接口，按照分数倒序排列；
3) 定义新闻类，包括编号（int 类型）、标题、内容和点击数，实现 Comparable 接口，按照编号正序排列；
4) 定义测试类，分别创建两个学生对象、新闻对象，进行比较并输出结果。