第七章 抽象、接口、内部类-答案

一、abstract关键字

1.基础问答

题目要求：

请解释**abstract**关键字的作用是什么。

答案及解析：

`abstract` 是一个关键字，用于声明抽象类和抽象方法。

抽象类是一种特殊的类，不能被实例化，只能被继承。它的作用是为子类提供一个通用的模板，子类必须实现抽象类中声明的抽象方法。抽象类可以包含非抽象方法，也可以包含成员变量、静态变量、常量等。

抽象方法是一种没有实现的方法，只有方法的声明，没有方法体。抽象方法必须声明在抽象类中，不能被直接调用，只有在子类中被重写并实现才能被使用。抽象方法的作用是为子类提供一个必须实现的方法，以保证子类的一致性和规范性。

抽象类和抽象方法的主要作用是提高代码的可扩展性和可维护性，使得程序的结构更加清晰和易于理解。它们可以使得程序更加灵活，便于扩展新的功能，同时也可以强制规范子类的行为，减少程序出错的概率。

需要注意的是，抽象类不能被实例化，如果想要使用抽象类的功能，必须通过继承抽象类来实现。而抽象方法必须被子类实现，否则子类也必须声明为抽象类。

2.基础问答

题目要求：

请简述抽象类的特点。

答案及解析：

1. 抽象类不能被实例化，只能被继承。
2. 只要包含抽象方法的类，必须是抽象类。
3. 抽象类可以包含抽象方法和非抽象方法，抽象方法没有具体实现，而非抽象方法有具体实现。
4. 子类必须实现抽象类中声明的所有抽象方法，否则子类必须声明为抽象类。
5. 抽象类可以有构造方法，但不能被直接调用，只能被子类调用。
6. 抽象类可以包含成员变量、静态变量、常量等，也可以实现接口。
7. 抽象类可以被用作多态的类型。

需要注意的是，抽象类并不是必须要有抽象方法，它也可以没有抽象方法。抽象类的作用主要是为子类提供一个通用的模板，子类必须实现抽象类中声明的抽象方法。抽象类还可以包含非抽象方法，这些非抽象方法可以被子类直接继承或者覆盖。抽象类的设计可以使得程序更加灵活，便于扩展新的功能，同时也可以强制规范子类的行为，减少程序出错的概率。

3.基础问答

题目要求：

请回答抽象类及抽象方法可以被final修饰符修饰吗?

答案及解析：

抽象类和抽象方法不能被final修饰，抽象类就是要子类继承然后实现内部方法的。但是final修饰的类是不能再被继承和修改的，所以不能用final修饰。

4.程序分析题

题目要求：

请阅读并分析以下程序，判断以下代码是否正确，如果正确，输出什么信息？如果不正确，请说明原因。

```
abstract class A {
    abstract void show();
}

class B extends A {
    void show() {
        System.out.println("B");
    }
}

class Main {
    public static void main(String[] args) {
        A a = new B();
        a.show();
    }
}
```

答案及解析：合法。输出为：

```
B
```

5.编程题

题目要求：

编写一个抽象类 Shape，包含以下属性和方法：

- 属性：颜色 color。
- 方法：计算形状面积的抽象方法 area() 和返回颜色信息的普通方法 getColor()。

然后创建两个具体的子类 Rectangle（矩形） 和 Circle（圆形），实现 area() 方法并考虑它们各自的特点。

要求能够使用以下测试代码进行测试：

```java
public class Test1_5_ShapeTest {
    public static void main(String[] args) {
        Shape rectangle = new Rectangle("Red", 2.0, 3.0);
        Shape circle = new Circle("Blue", 5);
        System.out.println(rectangle.getColor());
        System.out.println("Area: " + rectangle.area());
        System.out.println();
        System.out.println(circle.getColor());
        System.out.println("Area: " + circle.area());
    }
}
```

答案及解析：

```java
abstract class Shape {
    protected String color;

    public Shape(String color) {
        this.color = color;
    }

    public abstract double area();

    public String getColor() {
        return "Color: " + color;
    }
}

class Rectangle extends Shape {
    protected double width;
    protected double height;

    public Rectangle(String color, double width, double height) {
        super(color);
        this.width = width;
        this.height = height;
    }

    @Override
    public double area() {
        return width * height;
    }
}

class Circle extends Shape {
    protected double radius;

    public Circle(String color, double radius) {
```

```
        super(color);
        this.radius = radius;
    }

    @Override
    public double area() {
        return Math.PI * radius * radius;
    }
}
```

6.编程题

题目要求：

设计一个简单的动物园系统，包含以下几种动物：猫、狗、鸟。每种动物都有自己的特点和行为，但都有共同的属性和方法。其中，猫、狗和鸟都属于动物类，但它们的叫声和行为各不相同。

要求：

1. 使用抽象类来实现动物园系统，包含一个抽象方法 `makeSound()` 用于输出动物的叫声。
2. 每种动物都有一个属性 `name` 表示动物的名称，以及一个方法 `move()` 用于输出动物的行为。
3. 猫类具有属性 `color` 表示猫的颜色，重写 `makeSound()` 方法输出猫的叫声，并重写 `move()` 方法输出猫的行为。
4. 狗类具有属性 `age` 表示狗的年龄，重写 `makeSound()` 方法输出狗的叫声，并重写 `move()` 方法输出狗的行为。
5. 鸟类具有属性 `wingSpan` 表示鸟的翼展，重写 `makeSound()` 方法输出鸟的叫声，并重写 `move()` 方法输出鸟的行为。

要求能够使用以下测试代码进行测试：

```
class Test1_6_Zoo {
    public static void main(String[] args) {
        Animal cat = new Cat("tom", "灰白");
        Animal dog = new Dog("旺财", 5);
        Animal bird = new Bird("樱桃", 21.76);
        cat.makeSound();//猫发出喵喵叫声
        cat.move();//猫在四腿爬行
        dog.makeSound();//狗发出汪汪叫声
        dog.move();//狗在四腿奔跑
        bird.makeSound();//鸟发出啾啾叫声
        bird.move();//鸟在飞翔
    }
}
```

答案及解析：

```
// 定义抽象类 Animal
abstract class Animal {
    private String name;
```

```
4
5       public Animal(String name) {
6           this.name = name;
7       }
8
9       abstract void makeSound();
10
11      void move() {
12          System.out.println("动物在移动");
13      }
14  }
15
16  // 定义猫类 Cat
17  class Cat extends Animal {
18      private String color;
19
20      public Cat(String name, String color) {
21          super(name);
22          this.color = color;
23      }
24
25      @Override
26      void makeSound() {
27          System.out.println("猫发出喵喵叫声");
28      }
29
30      @Override
31      void move() {
32          System.out.println("猫在四腿爬行");
33      }
34  }
35
36  // 定义狗类 Dog
37  class Dog extends Animal {
38      private int age;
39
40      public Dog(String name, int age) {
41          super(name);
42          this.age = age;
43      }
44
45      @Override
46      void makeSound() {
47          System.out.println("狗发出汪汪叫声");
48      }
49
50      @Override
51      void move() {
52          System.out.println("狗在四腿奔跑");
```

```
    }
}

// 定义鸟类 Bird
class Bird extends Animal {
    private double wingSpan;

    public Bird(String name, double wingSpan) {
        super(name);
        this.wingSpan = wingSpan;
    }

    @Override
    void makeSound() {
        System.out.println("鸟发出啾啾叫声");
    }

    @Override
    void move() {
        System.out.println("鸟在飞翔");
    }
}
```

二、内部类

1.基础问答

题目要求：

请回答Java中的内部类有哪几种，并说明它们的特点。

答案及解析：

Java中的内部类有四种，具体如下：

- 成员内部类：定义在外部类的成员位置上，可以访问外部类的所有成员。
- 局部内部类：定义在方法内部的类，只能在方法内部访问，常用于简化复杂逻辑。
- 匿名内部类：没有名字的局部内部类，常用于实现接口或抽象类的匿名实现。
- 静态内部类：定义在外部类中的静态位置上，只能访问外部类的静态成员。

2.编程题

题目要求：

按照要求，补齐代码，要求在控制台输出 HelloWorld。

```
interface Inner {
    void show();
}

class Outer {
    //这里补全代码 提示：匿名内部类
}

class Test2_2_InnerClass{
    public static void main(String[] args) {
        Outer.method().show();
    }
}
```

答案及解析：

```
class Test2_2_InnerClass{
    public static void main(String[] args) {
        Outer.method().show();
    }
}

interface Inner {
    void show();
}

class Outer {
    //这里补全代码 提示：匿名内部类
    public static Inner method() {
        Inner inner = new Inner() {
            @Override
            public void show() {
                System.out.println("Hello World");
            }
        };
        return inner;
    }
}
```

3.编程题

题目要求：

使用匿名内部类方式实现下面接口，并完成规定的功能。

```
interface Action{
    public String[] test(String str);
}
```

对该接口进行实现，让其可以具有处理以下两种字符串的功能：

a.对字符串按照 | 进行分割，并得到数组，例如 传入字符串"a|b|c"，返回["a","b","c"]

b.对字符串按照 - 进行分割，并得到数组，例如 传入字符串"a-b-c"，返回["a","b","c"]

要求使用匿名内部类的方式来完成。

提示：**分割字符串，学习String类中split方法实现。**

答案及解析：

```
import java.util.Arrays;

class Test2_3_InnerClass {
    public static void main(String[] args) {

        //1.对字符串按照 | 进行分割，并得到数组
        //  例如 传入字符串"a|b|c"，返回["a","b","c"]
        Action ac1 = new Action() {
            @Override
            public String[] test(String str) {
                //字符串分割，按照'|'进行
                //  注意：'|' 是特殊字符，得专门处理：\\| 或 [|]
                //String[] array = str.split("[|]");
                String[] array = str.split("\\|");
                return array;
            }
        };

        String[] arr = ac1.test("a|b|c");
        System.out.println(Arrays.toString(arr));

        System.out.println("-----------------");

        //2.对字符串按照 - 进行分割，并得到数组，
        //  例如 传入字符串"a-b-c"，返回["a","b","c"]
        Action ac2 = new Action() {
            @Override
            public String[] test(String str) {
                //字符串分割，按照'|'进行
                //  注意：'-' 是普通字符，直接split("-")处理即可
                String[] array = str.split("-");
                return array;
            }
        };

        String[] arr2 = ac2.test("a-b-c");
        System.out.println(Arrays.toString(arr2));
    }
}
```

```
40
41 interface Action{
42     public String[] test(String str);
43 }
```

三、访问控制

1.基础问答

题目要求：

请描述各个权限修饰符的作用。

答案及解析：

- `public`：表明该成员变量或者方法是对所有类或者对象都是可见的，所有类或者对象都可以直接访问。
- `private`：表明该成员变量或者方法是私有的，只有当前类对其具有访问权限，除此之外其他类或者对象都没有访问权限，子类也没有访问权限。
- `protected`：表明成员变量或者方法对类自身，与同在一个包中的其他类可见，其他包下的类不可访问，除非是他的子类。
- `default`：注意，不是default关键字，而是不定义任何权限修饰符，这种级别被称为“default”级别。default级别表明该成员变量或者方法只有自己和其位于同一个包的内可见，其他包内的类不能访问,即便是它的子类。

四、interface

1.基础问答

题目要求：

请简述什么情况下该用抽象类、什么情况下该用接口。

答案及解析：

在Java中，抽象类和接口在不同的情况下可以有不同的应用场景，一般而言：

使用抽象类的情况：

1. 当多个类之间存在共同的行为和属性时，可以将这些共同的部分抽象成一个抽象类，供这些类进行继承，从而避免代码的重复编写。
2. 当需要在抽象类中定义一些具体方法（有实现的方法），这些方法可以在子类中直接使用或者覆盖，从而提供一些默认的实现。
3. 当需要在抽象类中定义实例变量，这些变量可以在子类中进行访问和使用。
4. 当希望在后续的版本中对类的设计进行扩展，添加新的方法或属性时，可以使用抽象类来实现。

使用接口的情况：

1. 当需要定义一组共同的行为，但这些行为可能与具体的类没有直接的关系时，可以使用接口来描述这些行为，从而实现代码的解耦和灵活性。
2. 当一个类需要实现多个不相关的行为（接口）时，可以通过实现多个接口来实现多重继承，因为Java不支持类的多重继承。
3. 当需要定义常量（静态final变量）时，可以使用接口来定义，因为接口可以包含常量的定义。

4. 当希望在后续的版本中对行为进行扩展，并且可能有多个类需要实现这些行为时，可以使用接口来实现。

需要注意的是，抽象类和接口都应该根据具体的需求和设计目标来选择使用，没有一种绝对的标准答案。在设计Java程序时，应该根据具体的场景和需求，合理地选择使用抽象类或接口，以达到代码的灵活性、可维护性和可扩展性。

2.编程题

题目要求：

根据以下描述编写程序。

鸟类（Bird）可以飞翔，飞机类（Plane）也可以飞翔，那么可以把飞翔（flyAction）这个动作抽象出来，抽象到一个抽象类或者接口中，然后让鸟类和飞机类去继承或实现，那么这里是把方法抽象到父类型中合适，还是抽象到接口中合适？编写代码完成本题。

要求能够使用以下测试代码进行测试：

```
class Test4_2_Interface {
    public static void main(String[] args) {
        Fly bird = new Bird();
        bird.flyAction(); //鸟会飞行
        Fly plane = new Plane();
        plane.flyAction(); //飞机能飞行
    }
}
```

答案及解析：

将飞翔（flyAction）这个动作抽象出来，抽象到一个接口中更为合适。因为飞翔这个行为与鸟类（Bird）和飞机类（Plane）之间没有直接的继承关系，而且鸟类和飞机类可能还会有其他不同的行为和属性。

```
interface Fly{
    void flyAction();
}

class Bird implements Fly{
    @Override
    public void flyAction() {
        System.out.println("鸟会飞行");
    }
}

class Plane implements Fly{
    @Override
    public void flyAction() {
        System.out.println("飞机能飞行");
    }
}
```

3.编程题

题目要求：

按照以下要求完成编程题。

```
定义一个Person类
    属性：name age
    行为：void run();

再定义一个Student类
    属性：id score
    行为：void study();

给Student类增强功能
    功能1：唱歌       接口 ISingAble { void sing();}
    功能2：飞行       接口 IFlyAble{ void fly(); }
```

答案及解析：

```
interface ISingAble {
    public abstract void sing();
}

interface IFlyAble{
    void fly();
}

class Person {
    private String name;
    private int age;

    public Person() {
        super();
    }
    public Person(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }

    public void run() {
```

```
        System.out.println(name + " 在奔跑");
    }

    @Override
    public String toString() {
        return "Person [name=" + name + ", age=" + age + "]";
    }
}

class Student extends Person implements ISingAble, IFlyAble {
    private String id;
    private double score;

    public Student() {
        super();
    }
    public Student(String name, int age, String id, double score) {
        super(name,age);
        this.id = id;
        this.score = score;
    }

    @Override
    public void fly() {
        System.out.println("学生 " + super.getName() + " 学会开飞机飞翔");
    }

    @Override
    public void sing() {
        System.out.println("学生 " + super.getName() + " 会唱歌");
    }
}
```

```
public class Test4_3_Interface {
    public static void main(String[] args) {
        //1.子类对象 赋值给 父类引用
        Person p = new Student("jack", 20, "002", 89.6);
        p.run();

        if (p instanceof Student) {
            Student s = (Student) p;
            s.fly();
            s.sing();
        }

        System.out.println("------------------");

        //2.子类对象 赋值给 接口引用
        ISingAble is = new Student("lucy", 21, "012", 80.6);
```

```
        is.sing();
        if (is instanceof Student) {
            Student s = (Student) is;
            s.fly();
            s.run();
        }

        System.out.println("-----------");

        if (is instanceof Student) {
            IFlyAble ifly = (IFlyAble) is;
            ifly.fly();
        }
    }
}
```

五、包装类

1.基础问答

题目要求：

请写出基本数据类型对应的的包装类名。

答案及解析：

基本数据类型	包装类
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean
char	Character

2.程序分析题

题目要求：

请阅读并分析以下程序案例，写出程序运行输出的结果。

```
Integer i1 = new Integer(97);
```

```
Integer i2 = new Integer(97);
System.out.println(i1 == i2);
System.out.println(i1.equals(i2));

Integer i3 = new Integer(148);
Integer i4 = new Integer(148);
System.out.println(i3 == i4);
System.out.println(i3.equals(i4));

Integer i5 = 97;
Integer i6 = 97;
System.out.println(i5 == i6);
System.out.println(i5.equals(i6));

Integer i7 = 148;
Integer i8 = 148;
System.out.println(i7 == i8);
System.out.println(i7.equals(i8));

int a = 148;
int b = 97;
Integer c = 97;
System.out.println(i7 == a);
System.out.println(b == c);
```

答案及解析：

```
Integer i1 = new Integer(97);
Integer i2 = new Integer(97);
System.out.println(i1 == i2);       // false
System.out.println(i1.equals(i2));  // true
```

```
Integer i3 = new Integer(148);
Integer i4 = new Integer(148);
System.out.println(i3 == i4);       // false
System.out.println(i3.equals(i4));  // true
```

```
Integer i5 = 97; // 自动装箱，从缓存区中取
Integer i6 = 97; // 自动装箱，从缓存区中取
System.out.println(i5 == i6);       //true
System.out.println(i5.equals(i6));  //true
```

```
Integer i7 = 148; // 自动装箱，未被缓存
Integer i8 = 148; // 自动装箱，未被缓存
System.out.println(i7 == i8);       //false
System.out.println(i7.equals(i8));  //true
```

```
int a = 148;
int b = 97;
Integer c = 97;
// c被自动拆箱，比较两基本数据类型数值大小
System.out.println(i7 == a);            //true
System.out.println(b == c);             //true
```

六、Object类

1.基础问答

题目要求：

请简述==和equals()的区别。

答案及解析：

== 是一个操作符，不能重写，而且大多被用在基础类型的比较上，因为基础类型不能使用equals方法。

== 操作符的本意就是用来比较操作符两边的对象是否是同一个，也就是他们的引用地址一不一样，速度很快。

而equals是个方法，来自于**java.lang.Object**类。在JAVA中，所有的类都是Object的子类，故所有对象都拥有equals方法。

如果不重写这个方法，那么默认使用的就是Object中的方法，源码如下：

```
public boolean equals(Object obj) {
    return (this == obj);
}
```

但是由于 == 是操作符不能重写，equals是方法可以重写，所以就产生了区别。所以 == 就永远比较的是对象地址，而equals在不重写的情况下也是的。其实他们两最大的区别就是使用目的的不同，== 主要用于基础类型的比较，只要基础类型的值相等那么就是true，在比较引用的时候则是比较引用的地址。equals呢则是主要用在引用对象的比较上，不重写的情况下和==一样，重写了就看自己的需要了。

2.程序分析题

题目要求：

请阅读并分析下面程序的运行结果。

```
class A {
}

class B extends A {
    void test() {
        System.out.println(super.getClass().getSimpleName());
        System.out.println(this.getClass().getSimpleName());
    }

    public static void main(String[] args) {
        new B().test();
    }
}
```

答案及解析：

```
B
B
```

因为子类与父类都没有重写getClass()方法，所以无论super.getClass()还是this.getClass()，结果都是一样的，调用到的都是Object中的getClass方法。

getClass()方法的作用是返回当前对象在运行时的类型，所以无论是通过super还是this调用该方法，获取到的都是同一个B类对象的类镜像。

七、String字符串比较问题

1.程序分析

题目要求：

阅读并分析以下两个案例中代码的运行结果。

案例一：

```java
String str1 = "hello";
String str2 = "hello";
String str3 = new String("hello");
String str4 = new String("hello");
String str5 = "hellohello";
String str6 = str1 + str2;
System.out.println(str1 == str2);
System.out.println(str2 == str3);
System.out.println(str3 == str4);
System.out.println(str2.equals(str3));
System.out.println(str3.equals(str4));
System.out.println(str5.equals(str6));
System.out.println(str5 == str6);
```

答案及解析：

```
true
false
false
true
true
true
false
```

案例二：

```java
String s1 = "helloworld";
String s2 = "hello";
String s3 = "world";
String s4 = "hello" + "world";
String s5 = s2 + s3;
String s6 = new String("helloword");
String s7 = "hello" + new String("world");
System.out.println(s1 == s4);
System.out.println(s1 == s5);
System.out.println(s1 == s6);
System.out.println(s1 == s7);
```

答案及解析：

```
true
false
false
false
```

2.基础问答

题目要求：

请回答String类型与基本数据类型与包装类如何转换?

答案与解析：