上海师范大学标准试卷

2023~2024 学年 第 2 学期　考试日期 2024 年 6 月 23 日

（考试时间：90 分钟）

科目：面向对象程序设计（A 卷）参考答案

一、判断题（每题 3 分，共 9 分）

1. Java 中的局部变量存在默认值　（ × ）
2. long 和 float 类型的常量需要加后缀　（ √ ）
3. 抽象类可以包含非抽象方法　（ √ ）

二、单选题（每题 3 分，共 9 分）

1. 以下哪个类型不是 Java 的基本数据类型　（ C ）

A. boolean　B. byte　C. String　D. float

2. 在 Java 中，类的默认修饰符是　（ D ）

A. public　B. protected　C. private　D. 无修饰符

3. FileInputStream 用于　（ A ）

A. 按字节读取文件　B. 按字符读取文件　C. 监视文件变化　D. 写入文件

三、填空题（共 12 分，1~2 题每题 3 分，第 3 题 6 分）

1. 只有__实例__方法才能调用实例变量。
2. Java 字节码文件的扩展名是__.class__。
3. 填写下表（填“√”或“×”）。

修饰符	同一个类	同一个包	子类（同包）	子类（不同包）	其他包
public	√	√	√	√	√
protected	√	√	√	√	×
默认 (无修饰符)	√	√	√	×	×
private	√	×	×	×	×

四、简答题（每题 6 分，共 30 分）

1. 什么是面向对象编程？请解释面向对象编程的三个主要特点。

面向对象编程（OOP）是一种编程范式，基于“对象”的概念，将程序组织成由数据和行为组成的对象。OOP 的三个主要特点是封装、继承和多态。

- 封装：将数据和操作封装在对象内部，只暴露必要的接口，隐藏内部实现细节。
- 继承：通过继承，子类可以复用父类的代码，并添加或修改功能。
- 多态：同一个接口可以有不同的实现，不同对象可以通过相同的接口调用不同的行为。

2. 简述 PATH 和 CLASSPATH 的配置方法及其区别。

PATH 的配置: 找到 JDK 的安装路径（例如：D:\jdk），设置环境变量，在 PATH 中添加 D:\jdk\bin

CLASSPATH 的配置: 设置 CLASSPATH 环境变量，.; D:\jdk\lib

3. Java 中的接口和抽象类有什么区别？请举例说明。

接口和抽象类的主要区别在于：

- 接口中只能有抽象方法和常量，抽象类可以包含具体方法和成员变量。
- 一个类可以实现多个接口，但只能继承一个抽象类。
- 接口用于定义行为规范，抽象类用于提供部分实现。

```
interface Animal {
    void makeSound();
}
abstract class Mammal {
    void breathe() {
        System.out.println("Breathing...");
    }
    abstract void makeSound();
}
class Dog extends Mammal implements Animal {
    void makeSound() {
        System.out.println("Barking...");
    }
}
```

4. 在 Java 中，什么是异常处理？请简要说明 try-catch-finally 块的使用及其重要性。

异常处理是 Java 中处理运行时错误的一种机制。通过异常处理，程序可以捕获和处理异常，避免程序崩溃。

try-catch-finally 块的使用：

- try 块：包含可能抛出异常的代码。
- catch 块：捕获并处理 try 块中抛出的异常。
- finally 块：包含总会执行的代码，无论是否抛出异常，常用于资源释放。

```
try {
    // 可能抛出异常的代码
} catch (Exception e) {
    // 处理异常
} finally {
    // 总会执行的代码
}
```

5. 请简要说明 Java 中的事件监听器（Event Listener）机制，并举例说明其在 GUI 编程中的应用。

事件监听器机制是 Java 处理用户交互的一种方式。在 GUI 编程中，通过事件监听器，可以响应用户的各种操作，如点击按钮、输入文本等。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class Example {
    public static void main(String[] args) {
        JFrame frame = new JFrame("Button Example");
        JButton button = new JButton("Click Me");
        button.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                System.out.println("Button clicked!");
            }
        });
        frame.add(button);
        frame.setSize(200, 200);
        frame.setLayout(new FlowLayout());
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
```

```
}
```

五、代码设计题（每题 10 分，共 40 分）

1. 编写一个名为 Box 的类，包含长度（length）、宽度（width）和高度（height）三个属性，并提供计算体积的方法。

```
public class Box {
    private double length;
    private double width;
    private double height;
    public Box(double length, double width, double height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }
    public double calculateVolume() {
        return length * width * height;
    }
    public static void main(String[] args) {
        Box box = new Box(2.0, 3.0, 4.0);
        System.out.println("Volume: " + box.calculateVolume());
    }
```

2. 编写一个名为 Animal 的抽象类，包含一个抽象方法 makeSound()。然后创建一个名为 Dog 的类继承该抽象类并实现 makeSound()方法。

```
abstract class Animal {
    abstract void makeSound();
}
class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Barking...");
    }
```

```
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.makeSound();
    }
}
```

3. 编写一个名为 Appliance 的接口，包含 turnOn、turnOff 和 getStatus 方法。然后创建一个名为 WashingMachine 的类实现该接口，并在 WashingMachine 类中重写这些方法，加入实际操作逻辑。同时，再创建一个名为 SmartWashingMachine 的类，继承自 WashingMachine 类，并重写 getStatus 方法，使其返回更加详细的状态信息。

```
interface Appliance {
    void turnOn();
    void turnOff();
    String getStatus();
}
class WashingMachine implements Appliance {
    private boolean isOn;
    @Override
    public void turnOn() {
        isOn = true;
        System.out.println("Washing machine is on.");
    }
    @Override
    public void turnOff() {
        isOn = false;
        System.out.println("Washing machine is off.");
    }
    @Override
    public String getStatus() {
        return isOn ? "Washing machine is running." : "Washing machine is off.";
    }
}
```

```
class SmartWashingMachine extends WashingMachine {
    @Override
    public String getStatus() {
        return super.getStatus() + " (Smart status)";
    }
    public static void main(String[] args) {
        SmartWashingMachine swm = new SmartWashingMachine();
        swm.turnOn();
        System.out.println(swm.getStatus());
        swm.turnOff();
        System.out.println(swm.getStatus());
    }
}
```

4. 编写一个名为 Employee 的类，包含一个 name 属性和一个 greet 方法，在方法中使用 this 关键字打印出 name 属性。然后创建一个名为 Manager 的类继承 Employee 类，并在 Manager 类的 greet 方法中调用父类的 greet 方法并增加自己的行为。

```
class Employee {
    String name;
    Employee(String name) {
        this.name = name;
    }
    void greet() {
        System.out.println("Hello, my name is " + this.name);
    }
}
class Manager extends Employee {
    Manager(String name) {
        super(name);
    }
    @Override
    void greet() {
```

```
        super.greet();
        System.out.println("I am the manager.");
    }
    public static void main(String[] args) {
        Manager manager = new Manager("Alice");
        manager.greet();
    }
}
```