

# Java 面向对象程序设计

## 注意事项:

1. 答卷前, 考生务必将自己的姓名和准考证号填写在答题卡上。
2. 回答选择题时, 选出每小题答案后, 用铅笔把答题卡对应题目的答案标号涂黑。如需改动, 用橡皮擦干净后, 再选涂其它答案标号。回答非选择题时, 将答案写在答题卡上。写在本试卷上无效。
3. 考试结束后, 将本试卷和答题卡一并交回。请认真核对监考员在答题卡上所粘贴的条形码上的姓名、准考证号与您本人是否相符。

## 一、 单选题

1. 编译 Java Application 源程序文件将产生相应的字节码文件, 这些字节码文件的扩展名为 ( )。  
A. java                      B. .class                      C. html                      D. .exe
2. 不允许作为类及类成员的访问控制符的是 ( )。  
A. public                      B. private                      C. static                      D. Protected
3. 设  $x = 1, y = 2, z = 3$ , 则表达式  $y + = z - - / + + x$  的值是 ( )。  
A. 3                      B. 3.5                      C. 4                      D. 5
4. 为 AB 类的一个无形式参数无返回值的方法 method 书写方法头, 使得使用类名 AB 作为前缀就可以调用它, 该方法头的形式为 ( )。  
A. static void method()                      B. public void method()  
C. final void method()                      D. abstract void method()
5. 下列属于容器的组件有: ( )。  
A. JButton                      B. JPanel                      C. Canvas                      D. JTextArea
6. void 的含义: ( )。  
A. 方法体为空                      B. 定义的方法没有形参  
C. 定义的方法没有返回值                      D. 方法的返回值不能参加算术运算
7. 关于 Java 中异常的叙述正确的是: ( )。  
A. 异常是程序编写过程中代码的语法错误  
B. 异常是程序编写过程中代码的逻辑错误  
C. 异常出现后程序的运行马上中止  
D. 异常是可以捕获和处理的
8. 下面哪个不是 java 语言中的关键字? ( )。  
A. long                      B. sizeof                      C. instanceof                      D. Const

9. 在复选框中移动鼠标, 然后单击一选项, 要捕获所选项必需实现哪个接口? ( ).
- A. ActionListener    B. MouseListener    C. MouseMotionListern    D. ItemListener
10. 以下有关类的继承的叙述中, 正确的是: ( ).
- A. 子类能直接继承父类所有的非私有属性, 也可继承父类的私有属性
- B. 子类只能继承父类的方法, 不能继承父类的属性
- C. 子类只能继承父类的非私有属性, 不能继承父类的方法
- D. 子类不能继承父类的私有属性

## 二、填空题

1. 开发与运行 Java 程序需要经过的三个主要步骤为 \_\_\_\_\_、  
\_\_\_\_\_ 和 \_\_\_\_\_。
2. 在 Java 的基本数据类型中, char 型采用 Unicode 编码方案, 每个 Unicode 码占用  
\_\_\_\_\_ 字节内存空间, 这样, 无论是中文字符还是英文字符, 都是占用 \_\_\_\_\_ 字节  
内存空间。
3. 设  $x = 2$ , 则表达式  $(x++)/3$  的值是 \_\_\_\_\_。
4. 若  $x=5, y=10$ , 则  $x < y$  的逻辑值为 \_\_\_\_\_,  $x \geq y$  的逻辑值为 \_\_\_\_\_。
5. \_\_\_\_\_ 方法是一种仅有方法头, 没有具体方法体和操作实现的方法, 该方法必须在抽象  
类之中定义。\_\_\_\_\_ 方法是不能被当前类的子类重新定义的方法。
6. 创建一个名为 MyPackage 的包的语句是 \_\_\_\_\_; 该语句应该放在程序  
的位置为: \_\_\_\_\_。
7. 设有数组定义: `int MyIntArray[] = {10, 20, 30, 40, 50, 60, 70};`; 则执行以下几个语  
句后的输出结果是 \_\_\_\_\_。

```
int s = 0;
for (int i = 0; i < MyIntArray.length; i++)
    if (i % 2 == 1)
        s += MyIntArray[i];
System.out.println(s);
```

8. 在 Java 程序中, 通过类的定义只能实现 \_\_\_\_\_ 重继承, 但通过接口的定义可以  
实现 \_\_\_\_\_ 重继承关系。

## 三、写出下面程序的运行结果

- 1.
- ```
import java.io.*;
public class abc {
    public static void main(String args[]) {
        AB s = new AB("Hello!", "I love JAVA.");
    }
}
```

```


        System.out.println(s.toString());
    }
}

class AB {
    String s1;
    String s2;
    public AB(String str1, String str2) {
        s1 = str1;
        s2 = str2;
    }
    public String toString() {
        return s1 + s2;
    }
}

```

运行结果: \_\_\_\_\_

2.




```

import java.io.*;
public class abc {
    public static void main(String args[]) {
        int i, s = 0;
        int a[] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
        for (i = 0; i < a.length; i++)
            if (a[i] % 3 == 0) s += a[i];
        System.out.println("s=" + s);
    }
}

```

运行结果: \_\_\_\_\_

3.



```

import java.io.*;
public class abc {
    public static void main(String args[]) {
        System.out.println("a="+a+"\nb="+b)
    }
}

class SubClass extends SuperClass {
    int c;
    SubClass(int aa, int bb, int cc) {
        super(aa, bb);
        c = cc;
    }
}

```

```

class SubSubClass extends SubClass {
    int a;
    SubSubClass(int aa, int bb, int cc) {
        super(aa, bb, cc);
        A = aa + bb + cc;
    }
    void show() {
        System.out.println("a=" + a + "\nb=" + b + "\nc=" + c);
    }
}

```

运行结果：

---




---



---

4.



```

public class Test {
    public static void main(String[] args) {
        int x; // x is declared but not used meaningfully here
        int a[] = { 0, 0, 0, 0, 0, 0 };
        calculate(a, a[5]); // a[5] is 0 initially
        System.out.println("the value of a[0] is " + a[0]);
        System.out.println("the value of a[5] is " + a[5]);
    }

    static int calculate(int x[], int y) {
        for (int i = 1; i < x.length; i++)
            if (y < x.length)
                x[i] = x[i-1] + i;
        return x[0];
    }
}

```

运行结果：

---



---

#### 四、简答题 (每题 5 分, 共 15 分)

1. 简单叙述如何使一个 Java 应用程序运行起来, 在控制台方式下, 使用什么命令编译、运行程序。
2. 简要叙述什么是访问控制符, 列出 Java 中的访问控制符, 并说明各个访问控制符的控制权限。
3. Java 从 JDK1.1 开始引入了委托事件模型, 简述其所采用的事件处理机制。

## 五、 编写程序 (每题 10 分, 共 20 分)

1. 设计一个 Circle 类, 该类包括的属性有: 圆心坐标和圆的半径, 包括的方法有: 设置和获取圆的坐标的方法, 设置和获取半径的方法, 计算圆的面积的方法。另外编写一个 Test 类, 测试 Circle 类。
2. 编写一个 Applet 程序, 创建一个空标签以及一个标识为“开始”的按钮。当鼠标按下“开始”按钮时, 就在标签上显示出“你好”, 这是一个 Applet 程序的样子。(Note: The provided code uses JFrame, not Applet, but fulfills the functional description).

## 六、 参考答案

### 6.1 单选题

1. **正确答案: B**

Java 源程序文件 (.java) 经过编译后生成字节码文件, 其扩展名为 .class。

2. **正确答案: C**

static 是一个修饰符, 用于定义静态成员 (变量或方法), 它不属于访问控制符。访问控制符包括 public, private, protected 以及默认 (包) 访问权限。

3. **正确答案: A**

表达式为  $y += z - - / ++x$ 。初始值:  $x = 1, y = 2, z = 3$ 。

1.  $++x$ :  $x$  变为 2。
2.  $z--$ : 使用  $z$  的当前值 3 进行运算, 之后  $z$  变为 2。
3. 运算  $/$ :  $z - - / ++x$  变为  $\frac{3}{2}$ 。由于是整数除法, 结果为 1。
4. 运算  $+=$ :  $y += 1$  变为  $y = y + 1 = 2 + 1 = 3$ 。

最终  $y$  的值为 3。

4. **正确答案: A**

要使得方法可以通过类名直接调用 (AB.method()), 该方法必须是静态 (static) 的。题目要求无形式参数、无返回值 (void), 因此方法头为 static void method()。

5. **正确答案: B**

JPane (通常指 JPanel) 是 Swing 中的一个容器组件, 可以用来容纳其他组件。JButton、Canvas (AWT 组件)、JTextArea 是普通组件, 不是容器。

6. **正确答案: C**

void 关键字用在方法声明中, 表示该方法执行后不返回任何值。

7. **正确答案: D**

- A 和 B 错误: 异常是在程序运行时发生的错误或意外情况, 不是编译时的语法错误或逻辑错误 (逻辑错误通常指程序能运行但结果不符合预期)。
- C 错误: 异常发生后, 如果不进行捕获处理, 程序会中止; 但如果使用 try-catch 块捕获并处理了异常, 程序可以继续执行。
- D 正确: Java 提供了异常处理机制 (try-catch-finally) 来捕获和处理运行时异常。

8. **正确答案: B**

sizeof 是 C/C++ 中的运算符, 用于获取类型或变量的大小, 不是 Java 的关键字。long, instanceof 是 Java 关键字。const 是 Java 的保留字, 但当前未使用。

9. **正确答案: D**

在复选框 (Checkbox) 或列表 (List) 中选择一项时, 会触发 ItemEvent 事件。处理这种事件需要实现 ItemListener 接口, 并重写 itemStateChanged 方法。ActionListener 用于按钮点击, MouseListener 和 MouseMotionListener 用于鼠标事件。

10. 正确答案：D

- A 错误：子类不能继承父类的私有 (private) 属性和方法。
- B 和 C 错误：子类继承父类的非私有属性和方法。
- D 正确：类的私有成员（属性和方法）只能在定义它们的类内部访问，不能被子类继承。

## 6.2 填空题

1. “编辑源程序”，“编译生成字节码”，“解释运行字节码”
2. “2”，“2”
3. “0”，“true”，“false”
4. “true”，“false”
5. “abstract”，“final”
6. “package MyPackage;”，“应该在程序第一句”
7. “120”
8. “单”，“多”

## 6.3 写出下面程序的运行结果

1. 运行结果: Hello!! love JAVA.
2. 运行结果: s=180  
a=60  
b=20  
c=30
3. 运行结果: the value of a[0] is 0  
the value of a[5] is 5
4. 运行结果: the value of a[0] is 0  
the value of a[5] is 5

## 6.4 简答题

1. 要运行一个 Java 应用程序，需要执行以下步骤：
  1. 编写源代码：使用文本编辑器编写 Java 源代码，并将其保存为以 .java 为扩展名的文件（例如 MyProgram.java）。文件名通常需要与文件中的公共类名（public class）相同。
  2. 编译源代码：打开控制台（命令行界面），使用 Java Development Kit (JDK) 中的编译器 javac 来编译源代码文件。命令格式为：

```
javac MyProgram.java
```

如果编译成功，会生成一个或多个字节码文件（.class 文件，例如 MyProgram.class）。如果存在语法错误，编译器会报告错误信息。

3. 运行程序：使用 Java 虚拟机 (JVM) 来解释执行生成的字节码文件。命令格式为：

```
java MyProgram
```

注意，在运行命令中不需要写 `.class` 扩展名。JVM 会加载 `MyProgram.class` 文件，并从 `main` 方法开始执行程序。

2. **访问控制符** (Access Modifiers) 是 Java 中的关键字，用于设定类、接口、构造函数、方法和变量（成员）的可访问范围（可见性），从而实现封装和保护。

Java 中的访问控制符及其权限如下：

| 访问控制符   | 关键字                    | 控制权限                                        |
|---------|------------------------|---------------------------------------------|
| 公共      | <code>public</code>    | 可以被任何其他类访问，无论它们在哪个包中。                       |
| 受保护     | <code>protected</code> | 可以被同一包中的所有类访问，以及不同包中的子类访问。                  |
| 默认（包私有） | （无关键字）                 | 只能被同一包中的类访问。如果类、方法或变量没有显式声明访问控制符，则默认为包访问权限。 |
| 私有      | <code>private</code>   | 只能在定义它的类内部访问。外部类（包括子类）无法直接访问。               |

Table 1: Java 访问控制符及其权限

3. Java 从 JDK 1.1 开始引入的**委托事件模型** (Delegation Event Model) 是处理 GUI（图形用户界面）事件和其他类型事件的核心机制。它基于三个主要组件：

- 1. **事件源 (Event Source)**：能够产生事件的组件对象。例如，一个按钮 (JButton) 是一个事件源，当用户点击它时，它会产生一个动作事件 (ActionEvent)。
- 2. **事件对象 (Event Object)**：封装了关于特定事件详细信息对象。它包含了事件的类型、事件源以及其他与事件相关的数据。例如，ActionEvent 封装了按钮点击事件的信息，MouseEvent 封装了鼠标操作的信息。事件对象都是 java.util.EventObject 的子类。
- 3. **事件监听器 (Event Listener)**：实现了特定监听器接口 (Listener Interface) 的对象。监听器接口定义了处理特定类型事件的方法。例如，要处理按钮点击事件 (ActionEvent)，需要实现 ActionListener 接口，并重写其 actionPerformed(ActionEvent e) 方法。

**事件处理机制流程：**

- 1. **注册监听器**：事件监听器对象必须向相应的事件源注册，表示它对该事件源产生的特定类型的事件感兴趣。例如，通过调用按钮的 addActionListener(listener) 方法来注册一个动作监听器。
- 2. **事件触发**：当用户与事件源交互（如点击按钮）时，事件源会创建一个相应的事件对象。
- 3. **事件派发（委托）**：事件源将创建的事件对象“派发”或“委托”给所有已注册的、能够处理该类型事件的监听器对象。
- 4. **事件处理**：监听器对象接收到事件对象后，调用其接口中定义的相应处理方法（如 actionPerformed），并传入事件对象作为参数。开发者在这些处理方法中编写具体的事件响应逻辑。



这种模型将事件的产生（由事件源负责）与事件的处理（由监听器负责）分离，使得代码结构更清晰，更易于维护和扩展。

## 6.5 编写程序

### 1. 参考代码:

下面是 Circle 类和 Test 类的参考实现:



```
// Circle.java
public class Circle {
    private int x;          // 圆心 x 坐标
    private int y;          // 圆心 y 坐标
    private double radius;  // 圆的半径 (使用 double 更通用)
    // private double v; // 面积可以实时计算, 不必作为成员变量存储

    // 构造函数
    public Circle(int x, int y, double radius) {
        this.x = x;
        this.y = y;
        // 半径不能为负数
        this.radius = (radius >= 0) ? radius : 0;
    }

    // --- Getter 方法 ---
    public int getX() {
        return x;
    }

    public int getY() {
        return y;
    }

    public double getRadius() {
        return radius;
    }

    // --- Setter 方法 ---
    public void setCoordinate(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public void setRadius(double radius) {
        // 半径不能为负数
        this.radius = (radius >= 0) ? radius : 0;
    }
}
```

```

// --- 计算面积的方法 ---
public double calculateArea() {
    // 使用 Math.PI 获取更精确的  $\pi$  值
    return Math.PI * radius * radius;
}

// (可选) 提供一个显示圆信息的方法
public void displayInfo() {
    System.out.println("圆心坐标: (" + x + ", " + y + ")");
    System.out.println("半径: " + radius);
    System.out.println("面积: " + calculateArea());
}
}

// Test.java
public class Test { // Test class name matches description
    public static void main(String args[]) {
        // 创建一个 Circle 对象, 圆心(2,5), 半径 6
        Circle c = new Circle(2, 5, 6.0);
        System.out.println("--- 初始圆信息 ---");
        c.displayInfo(); // 显示初始信息

        // 测试设置坐标
        c.setCoordinate(3, 4);
        System.out.println("\n--- 修改坐标后 ---");
        System.out.println("新圆心 X 坐标: " + c.getX());
        System.out.println("新圆心 Y 坐标: " + c.getY());

        // 测试设置半径
        c.setRadius(10.0);
        System.out.println("\n--- 修改半径后 ---");
        System.out.println("新半径: " + c.getRadius());

        // 测试计算面积
        System.out.println("\n--- 最终圆信息 ---");
        c.displayInfo(); // 显示最终信息
        // 或者直接获取面积
        // System.out.println("最终面积: " + c.calculateArea());
    }
}

```

## 2. 参考代码:

下面是满足要求的基于 Swing (JFrame) 的程序代码, 它创建了一个窗口, 包含一个标签和一个“开始”按钮。点击按钮时, 标签显示“你好”。(注意: 原始题目要求 Applet, 但提供的代码是基于 JFrame/Swing 的应用程序, 这里提供基于 Swing 的实现。)



```
import java.awt.FlowLayout; // Using a simpler layout manager
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.SwingUtilities; // For running GUI code safely

public class HelloAppletStyle extends JFrame implements ActionListener {

    private JLabel displayLabel;
    private JButton startButton;

    public HelloAppletStyle() {
        super("简单事件处理"); // Set window title

        // 1. Create components
        displayLabel = new JLabel(" "); // Start with an empty label (or
placeholder text)
        startButton = new JButton("开始");

        // 2. Set layout manager
        setLayout(new FlowLayout()); // Arrange components left-to-right

        // 3. Add components to the frame
        add(startButton);
        add(displayLabel);

        // 4. Register listener
        startButton.addActionListener(this); // 'this' object will
handle the button's actions

        // 5. Configure the frame
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // Close
operation
        setSize(300, 150); // Set window size
        setLocationRelativeTo(null); // Center the window
    }

    // 6. Implement the ActionListener interface method
    @Override
    public void actionPerformed(ActionEvent e) {
        // Check if the event source was the start button
        if (e.getSource() == startButton) {
            displayLabel.setText("你好"); // Set the label text as
required
        }
    }
}
```

```
// Main method to run the application
public static void main(String[] args) {
    // Run the GUI construction in the Event Dispatch Thread (EDT)
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            HelloAppletStyle app = new HelloAppletStyle();
            app.setVisible(true); // Make the window visible
        }
    });
}
```

**说明：**

- 这个程序创建了一个 JFrame 窗口。
- 使用 FlowLayout 自动排列按钮和标签。
- startButton 是事件源。
- HelloAppletStyle 类本身实现了 ActionListener 接口，成为事件监听器。
- actionPerformed 方法在按钮被点击时执行，将标签 displayLabel 的文本设置为“你好”。
- SwingUtilities.invokeLater 确保 GUI 的创建和更新在正确的线程（Event Dispatch Thread）中进行。