

## 目录

一、 选择题 (共 15 题, 每题 1 分, 共 15 分) .....	第 2 页
二、 判断题 (共 20 题, 每题 1 分, 共 20 分) .....	第 4 页
三、 程序分析题 (共 3 题, 共 20 分) .....	第 5 页
四、 简答题 (共 5 题, 共 30 分) .....	第 7 页
五、 编程题 (共 2 题, 共 15 分) .....	第 7 页
六、 参考答案 .....	第 9 页
6.1 选择题 .....	第 9 页
6.2 判断题 .....	第 18 页
6.3 程序分析题 .....	第 19 页
6.4 简答题 .....	第 20 页
6.5 编程题 .....	第 22 页

## 2024 — 2025 年《Java 程序设计》期末试题


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、 选择题 (共 15 题, 每题 1 分, 共 15 分)

1. 下列选项中, ( ) 是正确的 float 变量的声明。  
A. float f = 1;      B. float f = 1.0;      C. float f = 2e1;      D. float f = 2.3d;
2. 兼容性是 Java 语言的优势, 能做到这点是因为 Java 在操作系统的基础上提供了一个 Java 运行环境 ( ), 该环境由 Java 虚拟机 ( )、类库以及一些核心文件组成。  
A. JRE、JVM      B. JVM、JRE      C. JDK、JRE      D. JRE、JDK
3. 下列叙述中不正确的选项是 ( )  
A. 如果类里定义了一个或多个构造方法, 那么 Java 编译器不提供默认的构造方法。  
B. 同一个类创建的不同对象具有不同的实体。  
C. 不可以用一个类的对象访问类变量, 只能用类名访问类变量。  
D. 一个类的实例方法可以调用类中的其他实例方法和类方法。
4. 下列叙述中正确的选项是 ( )  
A. 可以用 protected 修饰一个类。  
B. protected 的访问权限低于友好的访问权限。  
C. byte 是一个基本的数据类型, 而 Byte 是一个类。  
D. 对象数组中每个空间存储的是对象的实体。
5. Java 语言中关于使用接口以下哪个说法正确? ( )  
A. 一个类只能实现一个接口。  
B. 一个非抽象类实现一个接口必须实现接口的所有方法。  
C. 接口之间不能继承。  
D. 接口和抽象类是同一回事。
6. 对于如下类, 下列叙述中正确的选项是 ( )  
A. 编译错误。  
B. 类中有 2 个构造方法, 2 个重载方法。

- C. 类中没有重载方法。
- D. 类中没有构造方法。



```
public class Test {  
    int x;  
    Test() {}  
    Test(int x) {}  
    void Test() {}  
    int Test(int m) {  
        x = m;  
        return x;  
    }  
}
```

7. 关于成员变量, 下列哪个叙述是正确的? ( )
- A. 成员变量的名字不可以和局部变量的名字相同。
  - B. 方法的参数的名字可以和方法中声明的局部变量的名字相同。
  - C. 成员变量没有默认值。
  - D. 局部变量没有默认值。
8. 下列叙述中正确的选项是 ( )
- A. 创建一个子类对象时, 对应的父类对象也一并创建。
  - B. 子类可以继承父类的构造方法。
  - C. 子类继承的方法不可以操作子类新声明的变量。
  - D. 子类新定义的方法没有办法操作子类隐藏的成员变量。
9. 以下哪个是正确的声明子类语句? ( )
- A. `class Student extend People {...}` 。
  - B. `class Student implements People {...}` 。
  - C. `class Student extends People {...}` 。。
  - D. `class Student implement People {...}` 。
10. 下列哪个叙述是错误? ( )
- A. `throws` 语句的作用是声明异常。
  - B. 在编写程序时可以扩展 `Exception` 类定义自己的异常类。
  - C. `try-catch` 语句可设由多个 `catch` 组成, `catch` 子句的处理与排列顺序无关。
  - D. `finally` 语句块中的代码总是被执行。
11. 下列变量定义中, 符合 Java 命名规范的是 ( )
- A. 3a
  - B. int name
  - C. \$number
  - D. field name
12. 为了区分重载多态中同名的不同方法, 要求 ( )

- A. 采用不同的参数列表。  
B. 返回值类型不同。  
C. 调用时用类名或对象名做前缀。  
D. 参数名不同。
13. 下列说法不正确的是 ( )。
- A. 一个 .java 源程序编译通过后, 得到的结果文件数也只有一个。  
B. 一个 .java 源程序编译通过后, 得到的文件的扩展名一定是 .class。  
C. 一个 .java 源程序只能有一个 public class 类定义, 且源文件的名字与 public class 的类名相同, 扩展名必须是 .java。  
D. 一个 .java 源程序可以包含多个 class 类。
14. 关于数组的叙述不正确的是 ( )。
- A. "int[] a;" 声明了一个 int 型一维数组。  
B. "int a[20];" 是正确的数组声明。  
C. 数组是引用型数据类型。  
D. 对于 "int a[][]=new int[2][9];", a.length 的值是 2。
15. 对于 "int n=6789;", 表达式的值为 7 的是 ( )。
- A.  $n \% 10$                       B.  $n / 10 \% 10$                       C.  $n / 100 \% 10$                       D.  $n / 1000 \% 10$

## 二、 判断题 (共 20 题, 每题 1 分, 共 20 分)

1. Java 经过编译产生的字节码是二进制代码, 可以直接在任何平台上识别和执行。 【 】
2. Java EE 主要用于嵌入式开发。 【 】
3. `byte x = (byte)(-129);` 该语句语法正确。 【 】
4. 类中方法体内声明的局部变量的有效范围为整个方法体内。 【 】
5. 若类 A 和 B 在不同包, 那么在类 B 中创建的类 A 对象不能访问类 A 的友好变量 【 】
6. 无包名的类可以使用有包名的类。 【 】
7. package 语句可以写在 Java 源程序中的任何位置。 【 】
8. this 可以出现在实例方法和构造方法中, 有时可能出现在类方法中。 【 】
9. super 关键字可以出现在子类的构造方法中任意位置。 【 】
10. StringTokenizer 类和 String 类的 split 方法都可以实现字符序列的分解。 【 】
11. 子类对象的上转型对象不可以调用子类新增的方法。 【 】
12. final 修饰类, 表示该类不可以有子类。 【 】
13. Java 支持多重继承, 即一个类可以继承多个类。 【 】
14. 所有类的根类是 Object 类。 【 】

15. 3e3 是 double 型常量。【 】
16. (byte)9 + 'c' 的结果是 int 型数据。【 】
17. 局部变量如果被 final 修饰就成为常量。【 】
18. Random 对象的 nextInt(int n) 方法随机返回 [0, n) 之间的一个整数。【 】
19. StringBuffer 对象的字符序列是不可以被修改的。【 】
20. FileReader 类可以实现以字节形式的文件的读取。【 】

### 三、程序分析题 (共 3 题, 共 20 分)

1. (6 分) 有以下源程序, 请写出【代码 1】的输出结果和理由, 【代码 2】和【代码 3】的错误理由。

程序 1:

```
class Father {
    int print(int x) {
        return x;
    }
}
class Son extends Father {
    int print(int x) {
        return x + 1;
    }
}
public class Test {
    public static void main(String args[]) {
        Father father = new Son();
        System.out.printf("%d", father.print(5)); // 【代码1】
    }
}
```

程序 2:


```
class E {
    int x; int n;
    n = 200; // 【代码2】
    public void f() {
        int m;
        int y = x + m; // 【代码3】
    }
}
```

【代码 1】结果及理由:

【代码 2】 错误理由:

【代码 3】 错误理由:

2. (6 分) 有以下源程序, 请写出下列 Test 类中【代码 1】、【代码 2】和【代码 3】的输出结果和理由。



```
public class Test {  
    public static void main(String args[]) {  
        String s1 = "Java程序设计";  
        String s2 = "Java" + "程序设计";  
        System.out.println(s1 == s2); // 【代码1】  
  
        String str = "Java";  
        String s3 = str + "程序设计";  
        System.out.println(s1 == s3); // 【代码2】  
  
        String s4 = new String("Java程序设计");  
        System.out.println(s3 == s4); // 【代码3】  
    }  
}
```

【代码 1】 结果及理由:

【代码 2】 结果及理由:

【代码 3】 结果及理由:

3. (8 分) 有以下源程序, 请写出 4 处标记错误的理由:

```
abstract class A {  
    int x = 2;  
    final abstract void method1(); // 【代码1】  
    static int method2() {  
        return x + 3; // 【代码2】  
    }  
}  
  
interface B {  
    public B() { // 【代码3】  
    }  
    public int method3(int a, int b) { // 【代码4】  
        return a + b;  
    }  
}
```

【代码 1】 错误理由:

【代码 2】 错误理由:

【代码 3】 错误理由:

【代码 4】 错误理由:

## 四、简答题 (共 5 题, 共 30 分)

1. (6 分) 简述访问权限中私有权限、公共权限、友好权限和受保护权限的特点。
2. (6 分) 简述对象组合及其优势。
3. (6 分) 简述面向抽象的编程方式。
4. (6 分) 列举 super 的两种常用场景。
5. (6 分) 分别简述什么是上转型变量、什么是接口回调。

## 五、编程题 (共 2 题, 共 15 分)

1. (共 6 分) 编写一个类 Sum, 包含两个方法, 其中一个方法为静态方法 getSum(), 其功能是求出  $2 + 4 + 6 + 8 + 10$  之和, 其值赋值给成员变量 sum; 另一个实例方法为 getMax(int

`x, int y, int z`), 其功能是求 `x, y` 和 `z` 的最大值, 其最大值赋值给成员变量 `max`。然后编写测试类 `Test`, 输出 `sum` 的值和最大值 `max`。

2. (共 9 分) 鸟类和昆虫类都具有飞行的功能, 要求利用接口完成以下功能:

- (1) 定义一个接口 `Fly`, 接口体中有个描述飞行的抽象方法 `fly()`。
- (2) 定义一个抽象父类 `Bird`, 父类中有个生蛋的抽象方法 `egg()`。
- (3) 定义鸽子类 `Pigeon`, 继承抽象父类 `Bird`, 实现接口 `Fly`。
- (4) 定义测试类 `Test`, 在主方法 `main` 中实例化 `Pigeon` 对象的上转型对象, 并使用多态的方法执行上转型对象的 `egg()` 方法。



## 六、 参考答案

### 6.1 选择题

#### 1. 正确答案：A

##### 1. A. `float f = 1;`

- 字面值 1 是整数 (int 类型)。在 Java 中, int 可以隐式转换为 float, 因为 float 有更大的数值范围 (尽管精度较低)。这是一个有效的声明。
- 正确。

##### 2. B. `float f = 1.0;`

- 字面值 1.0 在 Java 中默认为 double 类型 (所有没有后缀的浮点数字面值都是 double)。将 double 赋值给 float 需要显式转换 (如 `float f = (float) 1.0;` 或 `float f = 1.0f;`)。没有 f 后缀或类型转换会导致编译错误。
- 无效。

##### 3. C. `float f = 2e1;`

- 字面值 2e1 表示  $2 * 10^1 = 20.0$ , 默认为 double 类型 (科学计数法表示的数字除非加上 f 后缀, 否则都是 double)。和选项 B 一样, 将 double 赋值给 float 需要类型转换或加 f 后缀 (如 `float f = 2e1f;`)。没有后缀时这是无效的。
- 无效。

##### 4. D. `float f = 2.3d;`

- 字面值 2.3d 明确指定为 double 类型 (d 或 D 后缀表示 double)。将 double 赋值给 float 在没有类型转换的情况下是不允许的, 需要使用 f 后缀 (如 `float f = 2.3f;`)。这是无效的。
- 无效。

#### 2. 正确答案：A

##### 1. A. JRE、JVM

- JRE (Java Runtime Environment) 是 Java 运行环境的缩写, 包含 JVM (Java Virtual Machine) 和 Java 类库。JVM 是执行 Java 字节码的虚拟机。这个选项是正确的。
- 正确。

##### 2. B. JVM、JRE

- JVM 是 Java 虚拟机, JRE 是 Java 运行环境。这个选项的顺序是错误的。
- 无效。

##### 3. C. JDK、JRE

- JDK (Java Development Kit) 是 Java 开发工具包, 包含 JRE 和开发工具。这个选项的顺序是错误的。
- 无效。

##### 4. D. JRE、JDK

- JRE 和 JDK 是不同的概念, JDK 包含了 JRE。这个选项的顺序是错误的。
- 无效。

3. **正确答案：C**

1. A. 如果类里定义了一个或多个构造方法, 那么 Java 编译器不提供默认的构造方法。
  - 这是正确的说法。如果类中定义了任何构造方法, Java 编译器不会自动提供默认构造方法。
  - 正确。
2. B. 同一个类创建的不同对象具有不同的实体。
  - 这是正确的说法。同一个类创建的不同对象是不同的实体。
  - 正确。
3. C. 不可以用一个类的对象访问类变量, 只能用类名访问类变量。
  - 这是错误的说法。可以使用对象名或类名访问类变量 (静态变量), 但通常推荐使用类名来提高可读性。
  - 无效。
4. D. 一个类的实例方法可以调用类中的其他实例方法和类方法。
  - 这是正确的说法。实例方法可以调用同一类中的其他实例方法和静态方法。例如:

```
public class Example {  
    // 静态方法 (类方法)  
    public static void staticMethod() {  
        System.out.println("静态方法");  
    }  
  
    // 实例方法  
    public void instanceMethod1() {  
        System.out.println("实例方法1");  
    }  
  
    // 另一个实例方法, 可以调用实例方法和静态方法  
    public void instanceMethod2() {  
        instanceMethod1();    // 调用实例方法  
        staticMethod();       // 调用静态方法  
    }  
}
```

- 正确。

4. **正确答案：C**

1. A. 可以用 `protected` 修饰一个类。在 Java 中, `protected` 访问修饰符不能用于顶层类, 原因如下:
  1. 包控制: 顶层类应该要么是 `public` (在任何地方可见), 要么是包私有 (仅在包内可见)。这种设计使包的组织简单明了。

2. 继承目的: `protected` 专门为继承场景设计 – 它允许子类和同一包内的访问。由于顶层类不存在“可从包外继承但其他情况下隐藏”的用例, 所以在类级别使用 `protected` 没有意义。
3. 封装性: Java 设计者想要强制实施清晰的包边界。如果类只对子类可见但对其他类不可见, 会使包模型变得复杂。
4. 内部类例外: 注意 `protected` 可以用于内部类, 因为内部类是其封闭类的成员, 在这种情况下成员的可见性是有意义的。

这种设计使 Java 的访问控制模型简单明了, 并与面向对象原则保持一致。

- 无效。

2. B. `protected` 的访问权限低于友好的访问权限。

- 这是错误的说法。 `protected` 的访问权限高于友好的访问权限 (包内可见性)。  
`protected` 允许子类和同一包中的其他类访问。
- 无效。

3. C. `byte` 是一个基本的数据类型, 而 `Byte` 是一个类。

- 这是正确的说法。 `byte` 是 Java 中的基本数据类型, 而 `Byte` 是对应的包装类。
- 正确。

4. D. 对象数组中每个空间存储的是对象的实体。

- 这是错误的说法。在 Java 中, 对象数组存储的是对象的引用(为了性能), 而不是对象的实体。
- 无效。

修饰符	同类中	同包中	子类中	其他包
<code>private</code>	✓	✗	✗	✗
(default)	✓	✓	✗	✗
<code>protected</code>	✓	✓	✓	✗ (但子类可以访问)
<code>public</code>	✓	✓	✓	✓

表 1 Java 访问控制修饰符的可见性

5. **正确答案: B**

1. A. 一个类只能实现一个接口。

- 这是错误的说法。Java 允许一个类实现多个接口。
- 无效。

2. B. 一个非抽象类实现一个接口必须实现接口的所有方法。

- 这是正确的说法。如果一个非抽象类实现了一个接口, 它必须实现接口中定义的所有方法。
- 正确。

3. C. 接口之间不能继承。

- 这是错误的说法。接口可以继承其他接口。

- 无效。

4. D. 接口和抽象类是同一回事。

- 这是错误的说法。接口和抽象类在 Java 中是不同的概念，尽管它们都可以包含抽象方法：
  1. 抽象类可以有构造方法，接口不能
  2. 抽象类可以有非抽象方法，接口中的方法默认都是抽象的（Java 8 之前）
  3. 抽象类可以有实例变量，接口只能有常量
  4. 一个类只能继承一个抽象类，但可以实现多个接口

- 无效。

6. **正确答案：B**

1. A. 编译错误。

- 这是错误的说法。代码是有效的，编译器不会报错。
- 无效。

2. B. 类中有 2 个构造方法, 2 个重载方法。

- 这是正确的说法。类中有两个构造方法（Test()和 Test(int x)）和两个重载方法（void Test()和 int Test(int m)）。
- 正确。

3. C. 类中没有重载方法。

- 这是错误的说法。类中有两个重载方法（void Test()和 int Test(int m)）。
- 无效。

4. D. 类中没有构造方法。

- 这是错误的说法。类中有两个构造方法（Test()和 Test(int x)）。
- 无效。

```
public class Test {  
    int x;  
    Test() {}  
    Test(int x) {}  
    void Test() {}  
    int Test(int m) {  
        x = m;  
        return x;  
    }  
}
```

7. **正确答案：D**

1. A. 成员变量的名字不可以和局部变量的名字相同。

- 这是错误的说法。成员变量和局部变量可以同名，但在方法中使用时，局部变量会覆盖成员变量。
- 无效。

2. B. 方法的参数的名字可以和方法中声明的局部变量的名字相同。
- 这是正确的说法。方法参数和局部变量可以同名，但在方法中使用时，局部变量会覆盖方法参数。

但是参考答案说是错的

- 正确。
3. C. 成员变量没有默认值。
- 这是错误的说法。成员变量在类实例化时会被赋予默认值（如 int 类型默认为 0）。
  - 无效。
4. D. 局部变量没有默认值。
- 这是正确的说法。局部变量在使用前必须显式初始化，否则编译器会报错。
  - 正确。

```
public class Example {
    int memberVariable; // 成员变量

    public void method() {
        int localVariable; // 局部变量
        System.out.println(memberVariable); // 输出成员变量的默认值 (0)
        // System.out.println(localVariable); // 编译错误: 局部变量
        localVariable可能尚未初始化
    }
}
```

8. **正确答案：A**

1. A. 创建一个子类对象时，对应的父类对象也一并创建。
- 这是正确的说法。当创建子类对象时，父类对象会被隐式创建。
  - 正确。
2. B. 子类可以继承父类的构造方法。
- 这是错误的说法。子类不能继承父类的构造方法，但可以调用父类的构造方法。
  - 无效。
3. C. 子类继承的方法不可以操作子类新声明的变量。
- 这是错误的说法。子类可以在继承的方法中操作子类新声明的变量。
  - 无效。
4. D. 子类新定义的方法没有办法操作子类隐藏的成员变量。
- 这是错误的说法。子类新定义的方法可以操作子类隐藏的成员变量。
  - 无效。

概念	说明
构造方法继承	子类不能继承父类构造方法，但可以通过 <code>super()</code> 调用
对象创建顺序	创建子类对象时，先创建父类对象，再创建子类对象
变量访问	子类可以访问继承的变量和自己声明的变量
变量隐藏	子类可以声明与父类同名的变量，形成变量隐藏
方法重写	子类可以重写父类的方法，使用 <code>@Override</code> 注解
<code>super</code> 关键字	用于调用父类构造方法和访问父类成员

表 2 Java 继承机制的核心概念

9. **正确答案：C**

1. **A. `class Student extend People {...}` 。**
  - 这是错误的说法。Java 中使用 `extends` 关键字来继承类，而不是 `extend`。
  - 无效。
2. **B. `class Student implements People {...}` 。**
  - 这是错误的说法。`implements` 用于实现接口，而不是继承类。
  - 无效。
3. **C. `class Student extends People {...}` 。**
  - 这是正确的说法。Java 中使用 `extends` 关键字来继承类。
  - 正确。
4. **D. `class Student implement People {...}` 。**
  - 这是错误的说法。在 Java 中，`implement` 是错误的，应该是 `implements`。
  - 无效。

关键字	用途	示例
<code>extends</code>	用于类继承	<code>class Child extends Parent</code>
<code>implements</code>	用于实现接口	<code>class Child implements Interface</code>
<code>extends + implements</code>	同时继承类和实现接口	<code>class Child extends Parent implements Interface</code>
多重实现	实现多个接口	<code>class Child implements Interface1, Interface2</code>
单继承	只能继承一个类	Java 不支持 <code>class Child extends Parent1, Parent2</code>

表 3 Java 类继承与接口实现的语法规则

10. **正确答案：C**

1. **A. `throws` 语句的作用是声明异常。**
  - 这应该是错误的说法。😞😞(但是参考答案说是对的.大家以老师的答案为主)`throws` 用于声明方法可能抛出的异常，而不是声明异常本身。

- 无效。
2. B. 在编写程序时可以扩展 Exception 类定义自己的异常类。
- 这是正确的说法。Java 允许用户定义自己的异常类，通常通过扩展 Exception 类或其子类。
  - 正确。
3. C. try-catch 语句可设由多个 catch 组成, catch 子句的处理与排列顺序无关。
- 这是错误的说法。在 Java 中, catch 子句的顺序是重要的, 因为它们会根据异常类型进行匹配。
  - 无效。
4. D. finally 语句块中的代码总是被执行。
- 这是正确的说法。无论是否发生异常, finally 块中的代码都会被执行。
  - 正确。

```
public class Example {  
    public static void main(String[] args) {  
        try {  
            // 可能抛出异常的代码  
        } catch (Exception e) {  
            // 异常处理代码  
        } finally {  
            // 无论如何都会执行的代码  
        }  
    }  
}
```

11. 正确答案: C

变量名	说明
3a	错误: 变量名不能以数字开头
int name	错误: 变量名不能是 Java 关键字
\$number	正确: 变量名可以以美元符号开头
field name	错误: 变量名不可以包含空格

表 4 Java 变量命名规则

12. 正确答案: A

1. A. 采用不同的参数列表。
- 这是正确的说法。重载方法必须具有不同的参数列表（参数类型、数量或顺序）。
  - 正确。
2. B. 返回值类型不同。
- 这是错误的说法。返回值类型不能用于区分重载方法。

- 无效。
3. C. 调用时用类名或对象名做前缀。
- 这是错误的说法。调用重载方法时，使用对象名或类名并不会影响方法的选择。
  - 无效。
4. D. 参数名不同。
- 这是错误的说法。参数名不能用于区分重载方法，只有参数类型和数量可以。
  - 无效。

```
public class Example {  
    public void method(int a) {}  
    public void method(double b) {} // 重载：参数类型不同  
    public void method(int a, double b) {} // 重载：参数数量不同  
}
```

13. 正确答案：A

1. A. 一个 .java 源程序编译通过后，得到的结果文件数也只有一个。
  - 这是错误的说法。一个 .java 源程序可以包含多个类，每个类会生成一个 .class 文件。
  - 无效。
2. B. 一个 .java 源程序编译通过后，得到的文件的扩展名一定是 .class。
  - 这应该也是错误的说法。虽然大多数情况下是这样，但如果使用了其他工具或编译器，可能会生成不同类型的文件。(但是参考答案说是对的.大家以老师的答案为主)
  - 无效。
3. C. 一个 .java 源程序只能有一个 public class 类定义，且源文件的名字与 public class 的类名相同，扩展名必须是 .java。
  - 这是正确的说法。在 Java 中，一个 .java 文件只能有一个 public 类，并且文件名必须与该类名相同。
  - 正确。
4. D. 一个 .java 源程序可以包含多个 class 类。
  - 这是正确的说法。在 Java 中，一个 .java 文件可以包含多个非 public 类。
  - 正确。

14. 正确答案：B

1. A. "int[] a;" 声明了一个 int 型一维数组。
  - 这是正确的声明方式。在 Java 中，可以使用 int[] a; 或 int a[]; 声明一维数组。
  - 正确。
2. B. "int a[20];" 是正确的数组声明。
  - 这是错误的说法。在 Java 中不能在声明时指定数组大小。正确的方式应该是：
    - int[] a = new int[20]; 或
    - int a[] = new int[20];
  - 无效。



3. C. 数组是引用型数据类型。

- 这是正确的说法。在 Java 中，所有数组都是引用类型，即使是基本数据类型的数组。
- 正确。

4. D. 对于"int a[][]=new int[2][9];", a.length 的值是 2。

- 这是正确的说法。对于二维数组：
  - a.length 返回第一维的长度 (2)
  - a[0].length 返回第二维的长度 (9)
- 正确。

```
int[][] a = new int[2][9];
System.out.println(a.length);    // 输出: 2
System.out.println(a[0].length); // 输出: 9
```

15. 正确答案: C

1. A.  $n \% 10$

- $n \% 10$  得到 9 (取个位数)
- 无效。

2. B.  $n / 10 \% 10$

- $n / 10$  得到 678
- $678 \% 10$  得到 8 (取十位数)
- 无效。

3. C.  $n / 100 \% 10$

- $n / 100$  得到 67
- $67 \% 10$  得到 7 (取百位数)
- 正确。

4. D.  $n / 1000 \% 10$

- $n / 1000$  得到 6
- $6 \% 10$  得到 6 (取千位数)
- 无效。

## 6.2 判断题

题号					
1-5	✗	✗	✓	✓	✓
6-10	✓	✗	✗	✓	✓
11-15	✓	✓	✗	✓	✓
16-20	✓	✓	✓	✗	✗

Table 5 判断题答案一览表

1. 正确答案: ✗

Java 字节码 (.class 文件) 是二进制文件, 但不能直接在任何平台执行。它需要 JVM (Java 虚拟机) 来解释执行。

2. 正确答案: ✗

Java EE (Enterprise Edition) 主要用于企业级开发, 而不是嵌入式开发。Java ME (Micro Edition) 才是用于嵌入式开发。

3. 正确答案: ✗

虽然 -129 超出了 byte 类型的范围 (-128 到 127), 但是 (byte) 强制类型转换是合法的语法。转换结果会是 127。但是参考答案说是错的

4. 正确答案: ✓

局部变量的作用范围是从其声明位置开始到包含该声明的块结束, 而不是整个方法体。但是参考答案说是对的。大家以老师的答案为主。如以下情况:

```
public class Example {  
    public void method() {  
        int x = 10; // 局部变量x的作用范围从这里开始  
        if (x > 5) {  
            int y = 20; // 局部变量y的作用范围仅在if块内  
            System.out.println(x + y); // 可以访问x和y  
        }  
        // System.out.println(y); // 错误: y在这里不可见  
    }  
}
```

5. 正确答案: ✓

友好访问权限 (默认访问权限) 只允许同包内的类访问, 不同包的类无法访问。

6. 正确答案: ✓

无包名的类 (默认包) 可以使用有包名的类, 只要正确导入即可。

7. 正确答案: ✗

package 语句必须是 Java 源文件中的第一条非注释性语句。

8. 正确答案: ☒

this 只能在实例方法和构造方法中使用, 不能在静态方法 (类方法) 中使用。

9. 正确答案: ☒

在构造方法中, super()调用必须是第一条语句。

10. 正确答案: ☒

StringTokenizer 类和 String 类的 split()方法都可以用来分割字符串。

11. 正确答案: ☒

上转型对象只能访问父类中定义的方法, 不能直接调用子类新增的方法。

12. 正确答案: ☒

final 修饰的类不能被继承, 这是 Java 实现封装的一种方式。

13. 正确答案: ☒

Java 不支持多重继承, 一个类只能继承一个父类。但可以实现多个接口。

14. 正确答案: ☒

在 Java 中, 所有类都直接或间接继承自 Object 类。

15. 正确答案: ☒

科学计数法表示的浮点数字面量默认为 double 类型。

16. 正确答案: ☒

当 byte、short、char 类型的数据参与运算时, 会自动转换为 int 类型。

17. 正确答案: ☒

final 修饰的局部变量一旦初始化后就不能再改变其值, 成为常量。

18. 正确答案: ☒

nextInt(n)方法返回一个大于等于 0 且小于 n 的随机整数。

19. 正确答案: ☒

StringBuffer 对象的字符序列是可以修改的, 这是它与 String 类的主要区别。

20. 正确答案: ☒

FileReader 是字符流, 用于读取字符形式的文件, 而不是字节形式。

## 6.3 程序分析题

1.

代码 1	在这段代码中, Father 是父类, Son 继承并重写了 print 方法, 通过多态机制, 尽管变量类型是 Father, 但实际对象是 Son, 因此调用的是 Son 的 print 方法, print(5) 返回 6, 最终输出结果为 6。
代码 2	在 Java 中, 类的成员变量只能在构造方法或方法中初始化, 或者在声明时直接初始化。不能在类体中直接写赋值语句。应该改为 <code>int n = 200;</code> 或在构造方法中赋值。

代码 3	局部变量 m 在使用前未初始化。Java 要求局部变量在使用前必须显式初始化。应该先给 m 赋值，如 <code>int m = 0;</code> 或其他适当的值。
------	---

2.

代码 1	结果为 true。因为字符串字面量常量“Java 程序设计”和“Java”+“程序设计”在编译时就被优化为同一个字符串常量，它们指向字符串常量池中的同一个对象。
代码 2	结果为 false。因为 <code>str + “程序设计”</code> 涉及到运行时的字符串连接操作，会在堆内存中创建新的字符串对象，而不是使用字符串常量池中的对象。
代码 3	结果为 false。因为使用 <code>new String()</code> 会在堆内存中创建新的字符串对象，即使内容相同，也不是同一个对象。使用 <code>==</code> 比较的是对象的引用，而不是内容。

3.

代码 1	<code>final</code> 和 <code>abstract</code> 不能同时修饰一个方法。因为 <code>final</code> 方法不能被重写，而 <code>abstract</code> 方法必须被重写，这两个修饰符互相矛盾。
代码 2	静态方法不能直接访问非静态成员变量 x。因为静态方法属于类，而非静态变量属于对象实例。
代码 3	接口中不能定义构造方法。接口是一种规范，不能被实例化，因此不需要构造方法。
代码 4	接口中的方法默认是 <code>abstract</code> 的，不能有方法体。方法应该只有声明，实现由实现该接口的类来完成。

## 6.4 简答题

1. 访问权限特点

修饰符	同类中	同包中	子类中	其他包
<code>private</code>	✓	✗	✗	✗
(default)	✓	✓	✗	✗
<code>protected</code>	✓	✓	✓	✗ (但子类可以访问)
<code>public</code>	✓	✓	✓	✓

表 6 Java 访问控制访问权限特点

2. 对象组合

对象组合是一种设计原则，通过将一个对象嵌入到另一个对象来实现代码的复用和功能的扩展。组合比继承更灵活，可以在运行时动态地改变对象的组合关系。

优势：

- 提高代码复用性
- 降低类之间的耦合度
- 更灵活的对象关系

- 运行时可以动态改变组合关系

### 3. 面向抽象编程

- 针对接口或抽象类编程，而不是具体实现类
- 通过多态实现程序的可扩展性
- 遵循依赖倒置原则
- 提高代码的灵活性和可维护性

### 4. super 的常用场景

#### 1. 调用父类构造方法

```
public class Child extends Parent {  
    public Child() {  
        super(); // 调用父类构造器  
    }  
}
```

#### 2. 访问父类方法或属性

```
public class Child extends Parent {  
    public void method() {  
        super.parentMethod(); // 调用父类方法  
        super.parentField;    // 访问父类属性  
    }  
}
```

### 5. 上转型变量和接口回调

概念	说明
上转型变量	<ul style="list-style-type: none"><li>• 将子类对象赋值给父类类型的变量. 这种转换是安全的, 因为子类是父类的扩展, 包含父类的所有特性.</li><li>• 实现多态的基础, 允许在运行时决定调用哪个类的方法.</li><li>• 示例: <code>Animal animal = new Dog();</code></li></ul>
接口回调	通过接口来定义回调方法, 具体的实现由实现接口的类提供. 接口回调允许一个类调用另一个类的方法, 而不需要知道具体的实现细节.

表 7 Java 中的上转型变量和接口回调

```
// 回调接口示例  
interface Callback {  
    void onComplete(String result);  
}  
  
// 实现回调的类  
class Worker implements Callback {  
    public void onComplete(String result) {  
        // Other code...  
        System.out.println("任务完成: " + result);  
    }  
}
```

```

    }
}

// 使用回调的类
class Task {
    public void execute(Callback callback) {
        // 执行一些任务
        System.out.println("执行任务...");
        // 任务完成后调用回调方法
        callback.onComplete("处理完成");
    }
}

```

## 6.5 编程题

1.

```


// Sum类定义
class Sum {
    int sum;    // 成员变量sum
    int max;    // 成员变量max

    // 静态方法计算2+4+6+8+10的和
    public static int getSum() {
        return 2 + 4 + 6 + 8 + 10;
    }

    // 实例方法求三个数的最大值
    public int getMax(int x, int y, int z) {
        return Math.max(Math.max(x, y), z);
    }
}

// 测试类
public class Test {
    public static void main(String[] args) {
        Sum s = new Sum();
        s.sum = Sum.getSum();           // 调用静态方法计算和
        s.max = s.getMax(3, 8, 5);      // 调用实例方法求最大值

        System.out.println("sum = " + s.sum); // 输出和
        System.out.println("max = " + s.max); // 输出最大值
    }
}

```

2.

```


// 定义飞行接口
interface Fly {

```

```

    void fly(); // 抽象方法fly()
}

// 定义抽象父类Bird
abstract class Bird {
    abstract void egg(); // 抽象方法egg()
}

// 定义鸽子类
class Pigeon extends Bird implements Fly {
    @Override
    public void fly() {
        System.out.println("鸽子在飞行");
    }

    @Override
    public void egg() {
        System.out.println("鸽子在下蛋");
    }
}

// 测试类
public class Test {
    public static void main(String[] args) {
        Bird bird = new Pigeon(); // 创建上转型对象
        bird.egg(); // 调用egg()方法
    }
}

```