

2025 年春软件构造期末试题（回忆版）

闭卷考试，时间 120 分钟，满分 100 分。

一、 选择题（共 10 题。每题 2 分，共 20 分）

1. 下面选项哪个不是 Java 的语言特征（ ）
 - A. 平台无关
 - B. 面向对象设计
 - C. 支持显式指针
 - D. 垃圾回收机制
2. 对于 Java 继承的理解，正确的是（ ）
 - A. 子类可以继承父类的私有方法
 - B. 一个类可以同时继承多个接口
 - C. 继承会降低代码的可复用性
 - D. 子类必须重写父类的所有方法
3. 下面代码中，Program.main 的运行结果是哪一项（ ）

```
public interface IImpl {  
    int counter = 20;  
}  
  
public class Program implements IImpl {  
    private int counter = 0;  
    public static void main(String[] args) {  
        System.out.println(counter + 1);  
    }  
}
```

- A. 20
- B. 21
- C. 0
- D. 1

4. 下面对于白盒测试中条件覆盖的说法，正确的是哪一项（ ）
- A. 要求至少执行程序中所有语句一次
 - B. 至少执行程序中每个分支一次，保证程序中每个判定节点每种可能的结果至少一次
 - C. 保证程序中每个复合判定表达式中，每个简单判定条件的真假至少执行一次
 - D. 保证程序每个判定节点中，所有简单判定条件的所有可能的取值组合至少执行一次
5. 下面对于 Swing 容器的说法，错误的是哪一项（ ）
- A. `JFrame` 是一个顶层容器，包含标题、边框、菜单栏窗口等
 - B. `JDialog` 是一个顶层容器，用于创建对话框
 - C. `JApplet` 是一个类似小程序的顶层容器，现已逐渐弃用
 - D. `JPanel` 是一个轻量级容器，可以直接用来作为顶层窗口
6. 下面对于多线程的说法，正确的是哪一项（ ）
- A. 实现多线程需要调用 `Thread` 的 `run()` 方法
 - B. 通过实现 `Runnable` 接口来实现多线程的好处是，这个类还可以继承其他类
 - C. `Callable` 接口没有返回值，它和 `Runnable` 相同
 - D. 以上都不对
7. Java 语言编写网络通信程序一般位于 TCP/IP 的哪一个层级（ ）
- A. 应用层
 - B. 传输层
 - C. 网络层
 - D. 接口层
8. 下面对于 `HashSet` 的理解，正确的是哪一项（ ）
- A. 元素有序、可重复
 - B. 元素无序、不重复
 - C. 元素有序、无索引、可重复
 - D. 线程安全

9. Java 中用哪个类进行字符读入操作 ()
- A. InputStream
 - B. OutputStream
 - C. Reader
 - D. Writer
10. 下面对于数据访问对象模式的理解, 正确的是哪一项 ()
- A. 减少代码量
 - B. 隔离数据层, 把低级的数据访问操作从高级的业务服务中分离出来
 - C.
 - D.

二、 判断题 (共 5 题。每题 1 分, 共 5 分)

- 1. Swing 框架是 AWT 的轻量化替代, 其大部分组件由纯 Java 代码绘制而成, 不依赖本地平台组件。
- 2. 在 Java 中, 一个接口可以继承多个接口。
- 3. 在一个测试套组中, 如果实现了 100% 的语句覆盖, 则必然实现了 100% 的判定覆盖。
- 4. 迭代器模式的目的是为了提高集合的存储效率, 而不是提供一种统一的方法顺序访问一个聚合对象中的各个元素。
- 5. 工厂模式是一种行为型模式。

三、 填空题 (共 5 题。每空 1 分, 共 5 分)

- 1. 面向对象语言的三大特性是 _____、继承和多态。
- 2. 泛型通配符 <? _____ T> 指明传入类是 T 的子类。
- 3. 在 MVC 模式中, 模型与视图之间通过 _____ 模式交互。模型改变, 视图就会更新。
- 4. 按照目的来划分, 设计模式可以分为 _____、结构型模式和行为型模式。
- 5. TCP/IP 是最常用的网络协议。它基于 _____ (字节/字符) 流传输方式。

四、简答题（共6题。每题5分，共30分）

1. 多态如何定义？说出至少两种实现方法，并讲讲实现多态的优势有哪些。
2. 请叙述抽象类和接口的异同。（提示：请从属性、方法、继承、作用等方面进行比较。）
3. `ArrayList` 与 `LinkedList` 各自的特点都是什么？（提示：从底层结构看。）因此决定了它们应用的场景差异，这些差异有哪些？

4. 阅读以下代码，完成(1)~(3)问题。

```
public class DeadlockDemo {
    private static final Object lockA = new Object(), lockB = new
Object();
    public static void main(String[] args) throws InterruptedException
{
        Thread t1 = new Thread(() -> {
            synchronized (lockA) {
                System.out.println("T1: locked A");
                sleep(100);
                System.out.println("T1: trying to lock B");
                synchronized (lockB) {
                    System.out.println("T1: locked B");
                }
            }
        });
        Thread t2 = new Thread(() -> {
            synchronized (lockB) {
                System.out.println("T2: locked B");
                sleep(100);
                System.out.println("T2: trying to lock A");
                synchronized (lockA) {
                    System.out.println("T2: locked A");
                }
            }
        });
        t1.start();
        t2.start();
    }
    private static void sleep(long ms) {
        try {
            Thread.sleep(ms);
        } catch (InterruptedException ignored) { }
    }
}
```

(1) 代码产生什么问题？（1分）

(2) 这个问题如何产生？（2分）

(3) 怎么解决这个问题？（2分）

5. 简述关键字 `super` 与 `this` 的异同。

6. 阅读以下代码，完成(1)~(4)问题。

```
public class Buffer {
    private List<Integer> list = new ArrayList<>();
    private static final int MAX = 10;
    public void put(int value) {
        while (true) {
            try {
                Thread.sleep(500);
            } catch (InterruptedException ignored) { }
            synchronized (this) { // (A)
                while (list.size() == MAX) { // (B)
                    System.out.println("Buffer is full, waiting ....");
                    try { wait(); } catch (InterruptedException
ignored) { }
                }
            }
            System.out.println("Producer - " +
Thread.currentThread().getName() + " - put:" + value);
            list.add(value);
            notifyAll();
        }
    }
    public Integer take() { ... }
}
```

(1) 缓冲池为什么必须同步？（1分）

(2) 注释(A)处，**this**指代什么？（1分）

(3) 注释(B)处，如果把**while**改为**if**，会产生什么后果？（1分）

(4) 生产者 - 消费者模式的优点有哪些？（2分）

五、 综合题（共 5 题，其中 1 题 4 分； 2、 4 题 10 分； 3、 5 题 8 分。共 40 分）

1. 阅读以下代码，完成(1)~(3)问题。（4 分）

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

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

abstract class Animal { abstract void eat(); }

interface Run { void run(); }

class Dog extends Animal implements Run {
    @Override
    public void eat() { System.out.println("小狗吃骨头"); }
    @Override
    public void run() { System.out.println("小狗奔跑"); }
}

public class Test {
    public static void main(String[] args) {
        Student st = new Student("小王", 20, "哈工大");
        Dog dog = new Dog();
        System.out.print("同学来自" + st.school + "名字叫" + st.name + "
今年" + st.age + "岁");
    }
}
```


(1) 构造方法中使用 `super` 的作用是什么？（1 分）

(2) 语句 `Animal animal = new Animal();` 正确吗？为什么？（2 分）

(3) `Dog` 类继承 `Animal` 类并实现接口 `Run`，这是否违反 Java 标准？（1 分）

2. 在某系统中，存在一项“数据入库”任务。无论处理哪种类型的数据，都必须经过以下四个固定阶段。

[1] 数据加载

[2] 数据检验完整度

[3] 格式转化

[4] 数据写入

其中：阶段 1 和阶段 3 与输入的数据类型有关，因此需要对不同数据源分别实现；而其他阶段则是通用逻辑。基于上述背景，回答(1)~(4) 问题。（10 分）

(1) 什么是模板模式？（2 分）

(2) 联系多态，说说模板模式是如何实现不同类型数据的处理。（2 分）

(3) 画出模板模式的 UML 图（名称为 **AbstractClass** 以及 **ConcreteClass**）（4 分）

(4) 模板模式的优点有哪些？（2 分）

3. 某公司设计一套智能家居系统，功能要求如下：如果传感器（**Sensor**）检测到室外温度升高到一定温度，会将信号传递给响应设备，包括空调（**AirConditioner**）自动开启、窗帘（**Curtain**）自动闭合。为了支持将来引入新类型的相应设备，准备采用观察者模式设计该系统，回答(1)~(3)问题。（8分）

(1) 什么是观察者模式？（2分）

(2) 观察者模式的优缺点有哪些？（2分）

(3) 画出该系统的 UML 图。（4分）

4. 用策略模式实现快递运费的计算功能。寄标准快递（**BzStrategy**）的运费是货物价格的 5%（最低 10 块）；寄快递（**KdStrategy**）的基础运费是 5 块加上货物价格的 8%；寄特快快递（**TkStrategy**）的运费固定为 50 块。策略接口的代码已给出。（10 分）

```
public interface Strategy {  
    double Computing(double originalPrice);  
}
```

- (1) 策略模式的目的是什么？（2 分）

- (2) 通过以上接口实现三种快递方式的运费计算。（每个 2 分，共 6 分）

```
public class BzStrategy { ...  
  
}  
public class KdStrategy { ...  
  
}  
public class TkStrategy { ...  
  
}
```

- (3) 结合多态的概念，讨论在上述策略模式代码实现中，哪些地方使用了多态？（2 分）

