Java程 序 设 计

第 4 讲：流程控制与分支结构（if-else 与 switch）

厦门大学嘉庚学院

XIAMEN UNIVERSITY TAN KAH KEE COLLEGE

CONTENTS

1 程序结构与分支控制

2 随堂概念抢答与避坑找茬

3 switch 语句与现代语法增强

4 位运算与设备状态掩码 (Bitmask)

5 随堂编程小练笔与实操任务

6 课程总结与工程视野

程序设计的三大基本结构

【知识要点】结构化程序设计理论 (Böhm-Jacopini 定理)

任何复杂的算法逻辑都可以且仅由以下三种基本控制结构组合表达：

- **顺序结构**：按代码书写先后顺序自顶向下依次逐行执行；
- **选择/分支结构**：根据给定的条件表达式真假决定执行不同分支（`if-else`, `switch`）；
- **循环结构**：在满足特定条件时重复执行某段代码块（`for`, `while`, `do-while`）。

条件分支结构语法演进

单分支与双分支

```java
if (temp > 80.0) {
    alarm(); // 超温告警
}

if (isOnline) {
    sendPing();
} else {
    reconnect();
}
```

多分支阶梯结构

```java
if (score >= 90) {
    grade = 'A';
} else if (score >= 80) {
    grade = 'B';
} else if (score >= 60) {
    grade = 'C';
} else {
    grade = 'D';
}
```

【知识要点】重构原则:提前返回,消除深层嵌套

避免“箭头型”多层嵌套 if 代码;在方法入口处先做边界条件校验,不满足时直接 return,提升代码可读性与健壮性!

```java
// 推荐:扁平化卫语句写法
public void processPacket(byte[] data) {
    if (data == null || data.length == 0) return; // 守卫 1:空数据拦截
    if (data[0] != (byte)0xEB) return;            // 守卫 2:帧头不匹配拦截
    // 核心业务正常处理...
}
```

三目条件运算符与简洁表达式

【代码演练】三目运算符语法：条件? 表达式 1 : 表达式 2

```java
int a = 10, b = 20;
int max = (a > b) ? a : b; // 一行搞定取最大值

// 状态标签转换
String status = isOnline ? "在线" : "离线";
System.out.printf("设备状态: %s, 阈值: %d\n", status, max);
```

随堂抢答 1：悬挂 else（Dangling Else）陷阱

【随堂抢答·概念检测】问题：执行以下代码后，控制台输出什么内容？

```
int x = 5, y = 10;
if (x > 10)
    if (y > 5) System.out.print("A");
else
    System.out.print("B");
```

A. A

B. B

C. 无任何输出

D. 编译报错

随堂抢答 1：答案与深度解析

【知识要点】正确答案：【C 】

原理解析：

- Java 语法规定：`else` 总是与**离它最近的、尚未配对的** `if` 结合；
- 代码中的 `else` 实际上匹配的是内层的 `if (y > 5)`；
- 由于外层 `x > 10` 为 `false`，整个复合语句直接被跳过，**没有任何输出**！
- **规范**：永远显式书写大括号 `{ }`，严禁省略！

传统 switch 语句支持的数据类型

【知识要点】switch(expr) 允许的 6 种数据类型

- 基本整型：`byte`, `short`, `int`, `char`；
- 包装类型：`Byte`, `Short`, `Integer`, `Character`；
- 枚举类型：`enum`(Java 5+)；
- 字符串类型：`String`(Java 7+)；
- **注意**：`long`, `float`, `double`, `boolean` 均不能作为 switch 条件！

随堂找茬 1：switch 漏写 break 穿透陷阱

【随堂找茬·避坑挑错】看看以下代码传入 cmd = 1 时输出什么？

```
int cmd = 1;
switch (cmd) {
    case 1: System.out.print("启动 ");
    case 2: System.out.print("暂停 ");
    case 3: System.out.print("停止 ");
    default: System.out.print("未知");
}
```

【知识要点】穿透成因

漏写 `break;` 会导致程序从匹配的 case 开始一路无条件向下穿透执行所有后续分支！最终输出：" 启动暂停停止未知"！

Java 14+ 现代增强 switch 表达式

【代码演练】箭头表达式与无穿透语法 (ModernSwitchDemo.java)

```java
int cmd = 1;
// 现代 switch 箭头语法：天然杜绝穿透，支持直接返回值！
String status = switch (cmd) {
    case 1 -> "系统启动";
    case 2 -> "系统暂停";
    case 3 -> "系统停止";
    default -> "未知指令";
};
System.out.println("当前工作状态：" + status);
```

6 大位运算符真值表与通信掩码

运算符	运算规则	通信工程典型应用
& (按位与)	两位均为 1 时才为 1	**掩码提取特定状态位**
| (按位或)	只要有一位为 1 即为 1	**设置/置位特定功能位**
^ (按位异或)	两位不同为 1, 相同为 0	**特定位翻转 / CRC 校验**
~ (按位取反)	0 变 1, 1 变 0	清除特定标志位
<< (左移)	各二进制位左移, 低位补 0	快速乘 2^n / 报文高位拼接
>> (算术右移)	各二进制位右移, 高位补符号位	快速除 2^n / 报文截取

随堂抢答 2：位掩码提取运算预测

【随堂抢答·概念检测】问题：已知单字节状态码 status = 0b00001010，执行以下运算后结果是多少？

```java
byte status = 0b00001010; // 第 1 位与第 3 位为 1
byte mask   = 0b00000010; // 检测第 1 位（从 0 开始计数）
boolean isRunning = (status & mask) != 0;
```

A. true

B. false

C. 编译报错

D. 产生整型溢出

随堂抢答 2：答案与深度解析

【知识要点】正确答案：【A 】

原理解析：

- `0b00001010 & 0b00000010` 逐位进行逻辑与运算，结果为 `0b00000010`（十进制为 2）；
- `2 != 0` 为真，因此 `isRunning` 为 `true`；
- 这是通信协议中**单字节携带 8 个开关量状态**的标准高效解法！

随堂找茬 2：位运算符优先级陷阱

【随堂找茬·避坑挑错】看看以下代码为什么会发生编译报错？

```java
int status = 0x02;
int mask = 0x02;

// 编译报错: bad operand types for binary operator '&'
if (status & mask != 0) {
    System.out.println("设备运行中");
}
```

【知识要点】运算符优先级解析

关系运算符 `!=` 的优先级**高于**按位与 `&`！上述代码被 JVM 解释为 `status & (mask != 0)`，整数与布尔值做按位与导致类型不匹配！**必须强制加括号**：`(status & mask) != 0`！

随堂实操练笔 1：商场会员阶梯折扣结算器

【随堂实操·动手练笔】任务 1：编写 DiscountCalculator.java

编写 `DiscountCalculator.java`，实现：

1. 输入消费总金额与会员等级（1: 普通，2: 银卡，3: 金卡，4: 钻石）；
2. 钻石享 7 折且满 1000 再减 100，金卡享 8 折，银卡 9 折，普通原价；
3. 使用 `printf` 打印规范的消费明细账单。

随堂实操练笔 2:通信指令分发解析器

【随堂实操·动手练笔】任务 2:编写 ModernSwitchDemo.java

编写 `ModernSwitchDemo.java`,实现:

1. 接收十六进制命令字 `cmd`(如 `0x01`, `0x02`, `0x03`);
2. 使用现代 `switch` 表达式分发至“读取传感器”、“控制继电器”、“心跳握手”等分支;
3. 遇到非法指令时返回错误提示。

【工程视野】逻辑严密与分支覆盖

控制流是业务逻辑的核心；测试中追求“100% 分支覆盖率”，体现对系统质量与安全运行严密的工匠精神。

【实战技巧】代码审查与辅助开发

实战建议：请 AI 分析复杂分支代码的圈复杂度（Cyclomatic Complexity），并提供扁平化重构方案。

【随堂实操与课后思考】课后思考与巩固

- **练习**：使用位运算符实现设备 8 路开关量的独立置位（Set）、清零（Clear）与翻转（Toggle）。
- **思考**：在什么业务场景下，故意利用 switch 的穿透特性是合理的？

【知识要点】yield 在复杂代码块中返回值

在 Java 14+ 现代 switch 表达式中，如果某个 case 分支需要执行多行复杂逻辑，可在大括号中使用 `yield` 明确返回计算结果：

```
int mode = 2;
int bufferSize = switch (mode) {
    case 1 -> 1024;
    case 2 -> {
        System.out.println("启用大容量高吞吐通信缓冲模式");
        yield 4096; // 使用 yield 明确返回值
    }
    default -> 512;
};
```

随堂抢答 3：switch 表达式返回值语法

【随堂抢答·概念检测】问题：现代 switch 表达式中使用箭头语法时，以下哪个说法是错误的？

A. 天然杜绝 break 漏写导致的向下穿透

B. 支持直接给变量赋值

C. 每个 case 依然必须强制书写 break 语句

D. 支持逗号合并多个匹配项（如 case 1, 2, 3 ->）

【知识要点】正确答案：【C 】

原理解析：

- 现代 switch 箭头表达式（->）**天然杜绝穿透，绝对不需要也不能书写 `break;`！**
- 彻底消除了传统 switch 由于忘记写 break 导致的历史顽疾！

课程互动与讲师主页

在线主页与课程资源

- **个人主页**：
  `https://matnoble.top`
- **课程专栏**：
  `https://matnoble.top/courses/`
- **GitHub 仓库**：
  `https://github.com/MatNoble`

思考：Questions & Answers

“学贵善疑，逻辑筑基”

感谢大家的专注聆听！

课件最新版本与补充资料将持续同步至课程主页。

现在进入自由提问与交流时间。