

Java程 序 设 计

第 4 讲：流程控制与分支结构（if-else 与 switch）

- 1 本讲目标
- 2 循环一:让程序学会选择
- 3 循环二:switch 与穿透之谜
- 4 循环三:位运算与状态掩码(机动)
- 5 收束:自查与课后

本讲目标:下课前,你应该能做到

三件事

1. **会译**:把一条业务规则翻译成 if-else 或 switch 分支;
2. **会预判**:悬挂 else、switch 漏 break,行为说得出口;
3. **会用位**:用一个字节当 8 个开关,提取设备状态位。

课堂节奏

- 3 个循环,每个循环:讲 10 分钟 → 动手 5 分钟;
- 抢答先自己猜,再验证;实操限时完成,截图发课程群。

单分支 / 双分支

```
if (temp > 80.0) {  
    alarm();           // 超温才告警  
}  
  
if (isOnline) {  
    sendPing();  
} else {  
    reconnect();  
}
```

多分支阶梯

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

顺序、分支、循环——任何算法都由这三件东西搭成,本讲练分支。

卫语句:先排除异常,再谈正常流程

```
public void processPacket(byte[] data) {  
    if (data == null || data.length == 0) return; // 守卫1:空帧  
    if (data[0] != (byte) 0xEB) return;           // 守卫2:帧头不对  
    // 走到这里的一定是"干净"数据,专心写业务...  
}
```

- 反面教材:if 套 if 套 if 的“箭头代码”,读到第三层就迷路;
- 卫语句 = 提前 return,把异常拦在门口,主干保持扁平;
- 一行三目等价小事:int max = (a > b) ? a : b;。

抢答 1: 悬挂的 else 归谁?

【随堂抢答·概念检测】执行后输出什么?

```
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. 什么也不输出
- D. 编译报错

【知识要点】悬挂 else (Dangling Else)

- 规则:else 永远配给最近的、未配对的 if——即内层 if ($y > 5$);
- 外层 $x > 10$ 为假 → 整块跳过,B 根本没机会打印;
- 缩进会骗人,编译器不看缩进;
- 铁律:永远写大括号,哪怕里面只有一句。

实操 1: 会员阶梯折扣结算器

【随堂实操·动手练笔】DiscountCalculator • 限时 5 分钟

编写 DiscountCalculator.java:

1. 给定金额与等级(1 普通 / 2 银卡 9 折 / 3 金卡 8 折 / 4 钻石 7 折满千减百);
2. 用 if-else 阶梯计算应付金额;
3. printf 打印明细: 原价、折扣、应付。

完成标志: 金卡 800 应付 640.0 → 截图发课程群。

预测:漏写 break,输出什么?

【随堂找茬·避坑挑错】传入 cmd = 1,输出是?

```
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");
}
```

- A. 启动
- B. 启动暂停停止未知
- C. 未知
- D. 编译报错

【知识要点】switch 穿透(Fall-through)

- case 只是入口标签, break 才是出口;
- 漏写 break: 从匹配项起无视标签一路执行到底;
- switch 条件只认: byte short int char、String、枚举——long / double / boolean 不行;
- 顺序敏感的业务(如逐级审批)反而可故意利用穿透。

现代 switch: 箭头语法, 天然无穿透

```
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";  
};
```

- 箭头 ->: **只执行命中分支**, 没有穿透, 也不写 **break**;
- 它是**表达式**: 直接把结果赋给变量;
- 多行分支用大括号, 返回值写 `yield:yield 4096;`。

抢答 2: 箭头 switch, 哪个说法是错的?

【随堂抢答·概念检测】找出错误说法

- A. 天然杜绝漏 break 导致的穿透
- B. 支持直接给变量赋值
- C. 每个 case 依然必须强制书写 break
- D. 支持 case 1, 2, 3 -> 合并多个匹配项

【知识要点】新语法,新规矩

- 箭头分支不需要、也不允许 break——C 是旧习惯的残留;
- A、B、D 都是箭头 switch 的真实能力;
- 判断标准很简单:看见 -> 就当“一个分支一个值”。

【随堂实操·动手练笔】ModernSwitchDemo •限时 5 分钟

编写 ModernSwitchDemo.java:

1. 给定命令字:0x01 读传感器 / 0x02 控制继电器 / 0x03 心跳握手;
2. 用箭头 **switch** 表达式返回每条指令的动作描述;
3. 非法指令给出错误提示。

完成标志:三种合法指令各打一行、0x99 出提示 → 截图发课程群。

一个字节当 8 个开关用

运算符	规则	通信场景
& 与	全 1 才 1	提取状态位(掩码)
或	有 1 即 1	置位功能位
^ 异或	不同为 1	翻转 / CRC 校验
~ 取反	0 变 1	清除标志位
<< / >> 移位	左移右移	快速乘除 2^n 、拼报文

单字节携带 8 路开关量,是物联网协议的标准做法——第 17 讲的协议帧会大量用到。

抢答 3: 这位设备在运行吗?

【随堂抢答·概念检测】status 的第 1 位(从 0 数)是运行标志——结果是?

```
byte status = 0b00001010;  
byte mask   = 0b00000010; // 只看第 1 位  
boolean isRunning = (status & mask) != 0;
```

- A. true
- B. false
- C. 编译报错
- D. 整型溢出

【知识要点】按位与提取状态位

- $00001010 \ \& \ 00000010 = 00000010$ (十进制 2);
- $2 \neq 0 \rightarrow \text{true}$ ——其余 7 位全被掩码“遮黑”,只看想看的位;
- 姿势:先算 $\&$ 再比较,括号不能省。

找茬 2: 这行 if 为什么编译报错?

【随堂找茬·避坑挑错】报错: bad operand types for binary operator '&'

```
int status = 0x02, mask = 0x02;  
if (status & mask != 0) {  
    System.out.println("设备运行中");  
}
```

30 秒: 谁先算, 谁后算?

【知识要点】优先级陷阱

- 关系运算符 != 优先级**高于**按位与 &;
- 编译器读到的是 `status & (mask != 0)`: 整数 & 布尔 → 类型不匹配;
- 修复: `(status & mask) != 0`——**位运算永远亲手加括号**。

1. **会译**:折扣阶梯你用 if-else 写顺了吗?
2. **会预判**:悬挂 else 配谁? 漏 break 会怎样?
3. **会用位**:掩码提取、优先级括号——两个坑都踩过了?

没做到的那条,就是课后任务的第一优先级。

工程视野

分支写全 = 分支测全:追求 100% 分支覆盖率,是对系统质量与安全运行的严密承诺。

1. **练习**:用位运算对设备 8 路开关量做置位、清零、翻转(提示:|、& ~、^);
2. **AI 实战**:把你写的折扣器发给 AI,请它分析圈复杂度并给卫语句重构版,对比可读性。

下讲预告:循环——怎么让同一件事自动做一千次。

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>

自由交流与提问 (Q&A)

“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。