

第07课

天黑了，if 还是 else

让程序根据条件选择行动



从一扇条件门走到一条完整决策链：分清多条提醒与一条主建议，用边界值检查规则，最后完成能听懂回答的夜间生存顾问。

阅读方式：先猜结果，再运行核对；每次只改一个条件。

使用 Java 11 或更新版本。故事与插画借用方块世界，练习运行在普通 Java 控制台。

01 一扇没有墙的门



太阳刚躲到山后，史蒂夫便把门插在草地上：“安全屋，完工！” Alex 指指门两边：“墙呢？”一只僵尸绕过门走来，仿佛在称赞这座房子的通风设计。

程序也需要明确的门槛。天黑时做什么，生命低时做什么，两件事同时发生时先做什么？这一课，我们把含糊的“看情况”写成可以检查的规则。

- if 的条件成立时，哪些语句会执行？
- 提醒缺少的所有装备，与只选一个优先动作，写法相同吗？
- 生命恰好为0、8、9时，结果为什么不同？

生命0到20、补给门槛与建议顺序均为本课练习规则。输出“回屋”只是一行控制台文字，不会移动游戏角色。

02 if: 条件成立才行动

if 可以读作“如果”。圆括号里必须得到 boolean 值，花括号圈住条件成立时要做的事。程序执行到这里才判断一次；变量以后变化，不会让这段代码自动重演。

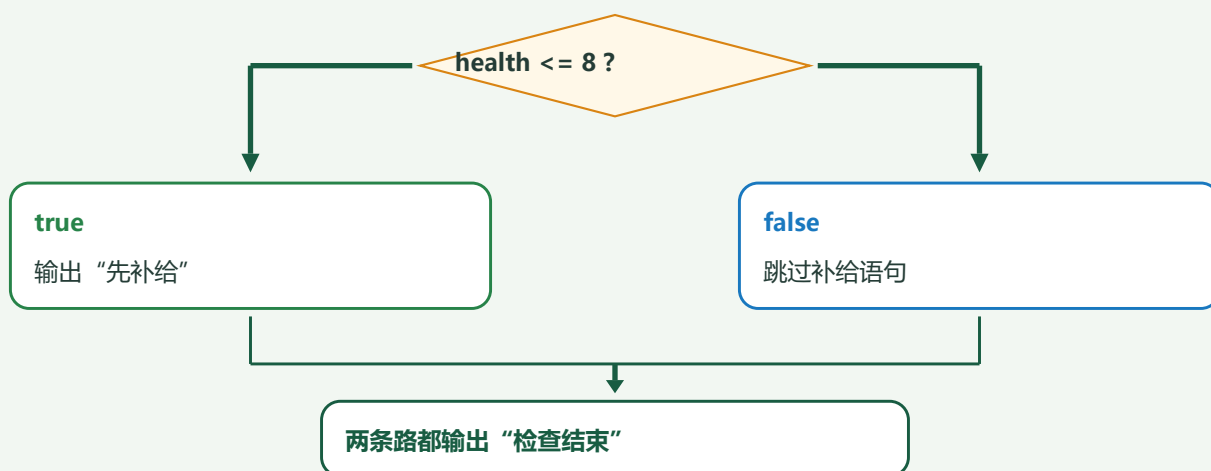
完整程序：保存为 HealthAlert.java

```
public class HealthAlert {  
    public static void main(String[] args) {  
        int health = 7;  
        if (health <= 8) {  
            System.out.println("先补给");  
        }  
        System.out.println("检查结束");  
    }  
}
```

运行核对

先补给
检查结束

条件门：两条路最后会合



把7改成12，只剩“检查结束”。后续标为“片段”的代码每次单独放进 main，不要把同名变量的例子粘成一大段。

03 else：否则走另一条路



想让条件不成立时也做一件事，就加上 else。同一次 if / else 判断只走一边，像矿车过岔道，不会复制出另一辆自己。

片段：放在 main 中，单独运行

```
boolean isNight = true;
if (isNight) {
    System.out.println("回屋休整");
} else {
    System.out.println("白天采集");
}
```

运行核对

回屋休整

片段：放在 main 中，单独运行

```
int diamonds = 2;
if (diamonds >= 3) {
    System.out.println("钻石数量够了");
} else {
    System.out.println("还差" + (3 - diamonds) + "颗钻石");
}
```

运行核对

还差1颗钻石

else 后不再写条件。把钻石改成3和4，都会报告数量够了；这里只检查钻石数量，还没有检查木棍或工作台。

04 else if: 顺序也是规则

一条分支链从上往下检查，选中第一个条件为真的分支，后面的分支便不再检查。先处理更紧急、更具体的情况，别让一个宽条件吞掉后面的窄条件。

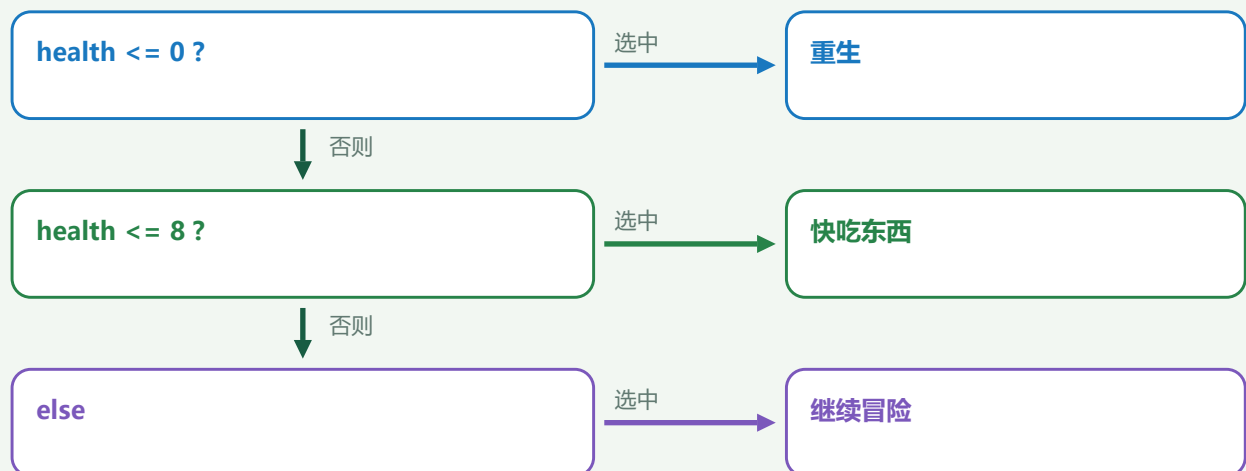
片段：放在 main 中，单独运行

```
int health = 7;  
if (health <= 0) {  
    System.out.println("重生");  
} else if (health <= 8) {  
    System.out.println("快吃东西");  
} else {  
    System.out.println("继续冒险");  
}
```

运行核对

快吃东西

只选一条：从上往下找入口



若先写 `health <= 8`，再写 `health <= 0`，生命0也会被第一条拦下。程序能编译，却会给出错误建议。这里暂研究顺序，稍后补上合法范围检查。

05 把小条件拼成大问题

先用一句话讲清楚条件

&& 并且

两项都满足：天黑并且没有住所。

|| 或者

至少一项满足：有钥匙或者是主人。

! 取反

把真假翻过来：!hasShelter 表示没有住所。

片段：放在 main 中，单独运行

```
boolean isNight = true, hasShelter = false;
if (isNight && !hasShelter) {
    System.out.println("先建临时住所");
}
```

运行核对

先建临时住所

片段：放在 main 中，单独运行

```
boolean hasKey = false, isOwner = true;
if (hasKey || isOwner) {
    System.out.println("门开了");
}
```

运行核对

门开了

片段：放在 main 中，单独运行

```
String weather = " RAIN ".strip();
if ("rain".equalsIgnoreCase(weather)) {
    System.out.println("把施工计划挪到屋内");
}
```

运行核对

把施工计划挪到屋内

&& 左侧为false、|| 左侧为true时，会跳过右侧计算。用括号写清组合规则，例如 `thunder || (isNight && !hasShelter)`。文字比较内容用 `equals` 或 `equalsIgnoreCase`。

06 多条提醒，还是一条主建议？

缺面包也缺盾牌，可以提醒两件事；决定“现在先做什么”，则可能只需要一句主建议。先确定任务，再选择独立 if 或分支链。

片段：放在 main 中，单独运行

```
int health = 6;
if (health <= 8) {
    System.out.println("带面包");
}
if (health <= 16) {
    System.out.println("带盾牌");
}
```

运行核对

带面包
带盾牌

片段：放在 main 中，单独运行

```
int health = 6;
if (health <= 8) {
    System.out.println("带面包");
} else if (health <= 16) {
    System.out.println("带盾牌");
}
```

运行核对

带面包

条件相同，组织方式不同

两个独立 if

两个条件各查一次，可能输出两条提醒。

一条 if / else if 链

选中前一条后，不再检查后面的条件。

把生命改成20，两段都没有输出。没有末尾 else 的分支链，允许一个分支也不执行。

07 Bug 最爱坐在门槛上

7能正确运行，不代表8也正确。先规定生命的合法范围，再把它分成没有遗漏、不会重复命中的几档。每个门槛都测试它自己和两边的邻居。

片段：放在 main 中，单独运行

```
int health = 8;
if (health < 0 || health > 20) {
    System.out.println("生命范围错误");
} else if (health == 0) {
    System.out.println("重生");
} else if (health <= 8) {
    System.out.println("低生命");
} else if (health <= 16) {
    System.out.println("谨慎前进");
} else {
    System.out.println("生命充足");
}
```

运行核对

低生命

生命分档：把测试点写在规则旁

输入范围	本课结果	代表测试值
范围外	生命范围错误	-1、21
0	重生	0
1到8	低生命	1、8
9到16	谨慎前进	9、16
17到20	生命充足	17、20

走到 `health <= 8` 时，范围外和0已经被排除；因此这一分支实际表示1到8。最后的 `else` 只剩17到20，不代表所有整数。

08 嵌套：进村后再找房间



先问是不是夜晚，再在夜晚分支里问有没有住所，这就是嵌套。缩进帮我们看层次，花括号才真正决定哪些语句属于哪一层。

片段：放在 main 中，单独运行

```
boolean isNight = true, hasShelter = false;
if (isNight) {
    if (hasShelter) {
        System.out.println("回屋休整");
    } else {
        System.out.println("先建临时住所");
    }
} else {
    System.out.println("白天采集");
}
```

运行核对

先建临时住所

片段：放在 main 中，单独运行

```
boolean isNight = false;
String advice;
if (isNight) {
    advice = "回家";
} else {
    advice = "采集";
}
System.out.println(advice);
```

运行核对

采集

advice 先在外面声明，两条路都给它赋值，块外才能读取。声明在某个分支里的局部变量，不能跨出那个代码块使用。

09 能运行，也可能判断错



逻辑反例：先预测，再找多余符号

```
boolean isNight = false;
if (isNight); { // 错误示范：多了分号
    System.out.println("关门");
}
```

运行核对

关门

分号是一条空语句。这里 if 只控制这条空语句，后面的代码块独立执行；删掉条件后的分号才能修好。

逻辑反例：钥匙被程序偷偷改了

```
boolean hasKey = false;
if (hasKey = true) { // 错误示范：赋值，不是比较
    System.out.println("开门");
}
```

运行核对

开门

boolean 赋值也会产生 boolean 结果，因此这个错误竟然能编译。改成 if (hasKey) 就够了；若写 if (health = 8)，得到的是 int，则不能作为条件。

- 核对花括号，避免 else 配错 if。
- 核对 = 与 ==，不要把赋值当成比较。
- 核对 < 与 <=，用恰好等于门槛的值测试。

10 先猜，再跑，最后解释

让每次运行都留下思考

先猜

写下输出与会命中的分支。



再跑

只运行当前片段，核对每一行。



解释

指出哪次判断改变了路线。

片段：放在 main 中，单独运行

```
int diamonds = 3, sticks = 1;
if (diamonds >= 3 && sticks >= 2) {
    System.out.println("材料齐了");
} else {
    System.out.println("材料还不齐");
}
```

片段：放在 main 中，单独运行

```
int health = 8;
if (health <= 8) {
    health += 2;
}
if (health > 8) {
    health += 1;
}
System.out.println(health);
```

核对答案

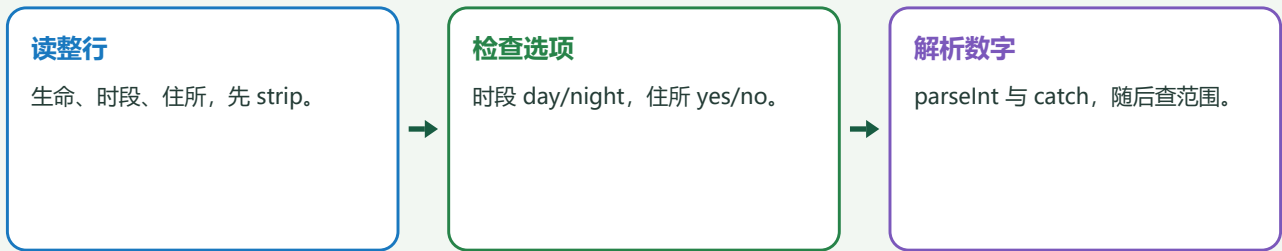
第一题输出“材料还不齐”，因为木棍只有1根。第二题输出11：第一次把8变成10，第二个独立 if 读到新值10，又加1。

只把第二题的第二个 if 改成 else if，结果变成10。前一支已经选中，后面的条件就不再检查。

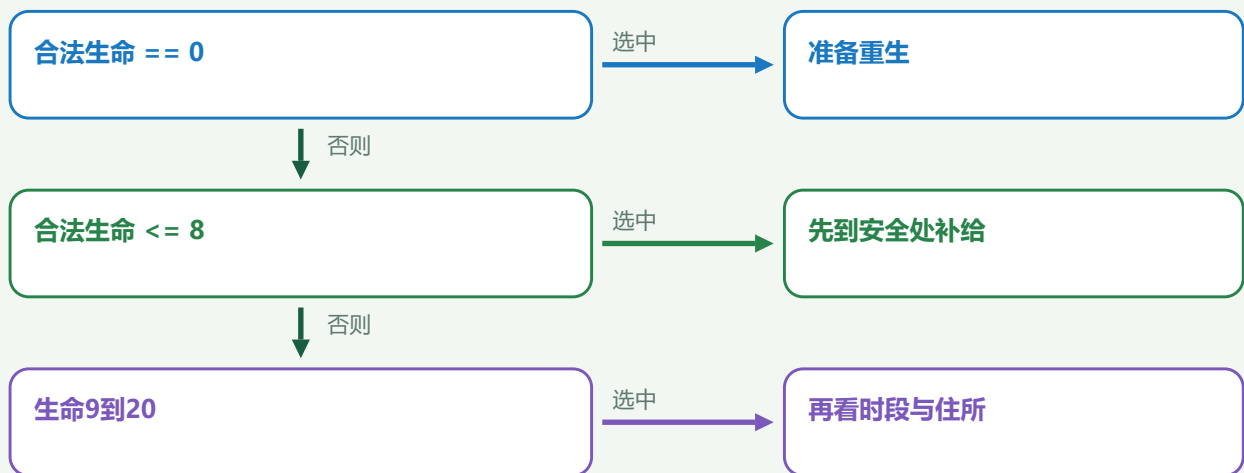
11 项目蓝图：夜间生存顾问

“别在我生命为0时推荐装修。”史蒂夫的需求很合理。这个顾问每次读三行输入，只给一条主建议；要再次询问，就重新运行程序。

先验输入，再作决定



主建议优先级：前面选中就停止向下检查



时段与住所有未知文字，先报告选项错误。生命不能解析则报告整数错误，越界则报告范围错误。nite 不能直接当白天：false 只有在二选一验证通过后才有明确含义。

12 顾问代码：读取与验证

新建 NightAdvisor.java。完整程序印在本页和下一页：按顺序连在同一个文件内，不要在两页之间额外添加花括号。

NightAdvisor.java • 上半部分，下一页继续

```
import java.util.Scanner;

public class NightAdvisor {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("生命 0-20: ");
        String healthText = in.nextLine().strip();
        System.out.println("时段 day/night: ");
        String time = in.nextLine().strip();
        System.out.println("有住所 yes/no: ");
        String home = in.nextLine().strip();
        boolean timeOK = "day".equalsIgnoreCase(time)
            || "night".equalsIgnoreCase(time);
        boolean homeOK = "yes".equalsIgnoreCase(home)
            || "no".equalsIgnoreCase(home);
        if (!timeOK || !homeOK) {
            System.out.println("选项有误，请检查时段和住所");
        } else {
            try {
                int health = Integer.parseInt(healthText);
                boolean isNight = "night".equalsIgnoreCase(time);
                boolean hasShelter = "yes".equalsIgnoreCase(home);
```

三个输入都用 nextLine 读取，避免混用 nextInt 时留下行尾。timeOK 与 homeOK 先确认词语合法；通过后才把 night、yes 转成 true。

试着手推：8 / night / yes 会先补给；9 / night / yes 会回屋。只差1点生命，优先级就决定了不同的路线。

13 顾问代码：选择与收尾

NightAdvisor.java • 紧接上一页

```
        if (health < 0 || health > 20) {
            System.out.println("生命必须在0到20之间");
        } else if (health == 0) {
            System.out.println("生命耗尽, 准备重生");
        } else if (health <= 8) {
            System.out.println("先到安全处补给");
        } else if (isNight) {
            if (hasShelter) {
                System.out.println("回屋休整");
            } else {
                System.out.println("先建临时住所");
            }
        } else {
            System.out.println("白天采集, 别忘带补给");
        }
    } catch (NumberFormatException e) {
        System.out.println("生命请输入0到20的整数");
    }
}
}
```

try/catch 沿用第6课的解析方法。若输入 abc、小数或装不进 int 的大数, catch 给出整数提示。输入合法后, 这条分支链只会输出一条主建议。

运行后依次输入8、night、yes, 每行按回车。三个提示之后应输出“先到安全处补给”。输出不是游戏动作, 而是顾问对练习数据的判断。

14 验收：一次只改一个输入

只动一个变量，观察路线变化

8 / night / yes

先到安全处补给

9 / night / yes

回屋休整

9 / night / no

先建临时住所

顾问测试表

生命 / 时段 / 住所	预期主结果
0 / day / yes	生命耗尽，准备重生
20 / day / no	白天采集，别忘带补给
-1或21 / day / yes	生命必须在0到20之间
abc、8.5或过大整数 / day / yes	生命请输入0到20的整数
9 / nite / yes	选项有误，请检查时段和住所
9 / night / maybe	选项有误，请检查时段和住所

让测试成为证据

- 再测1、8、9、20，确认每条主建议都被访问。
- 试两端空白和大写 NIGHT / YES，确认清理与忽略大小写生效。
- 生命和选项同时错误时，按约定先报告选项错误。
- 每行记录输入、预测、实际结果和差异原因。

15 把判断技能装进背包



木镐任务：钻石数量检查

先用变量，再接键盘输入。负数报错，至少3颗报告够用，否则算还差多少。测试-1、0、2、3、4；2应差1颗。

铁镐任务：生命四档广播

范围限定0到20，分成0、1到8、9到16、17到20，只输出一个档位。用九个边界测试值核对，并解释最后 else 为什么只剩最后一档。

钻石镐任务：天气与装备顾问

读取 clear、rain、thunder 与 yes/no。先验证，再处理雷雨与住所，最后才给普通天气建议。增加护甲条件时，先写清优先级，至少测试6组。

离开前能说清：独立 if 与分支链有什么区别？为什么0不能被 ≤ 8 接住？哪一组边界数据抓到了你的 Bug？下一课将用 switch 整理一组固定选项。

技术核对：Oracle Java 语言规范第14章，if、代码块与局部变量。
<https://docs.oracle.com/javase/specs/jls/se21/html/jls-14.html>