

第09课

循环建造机

让计数器记路，让累加器记收获



从五格石砖到二十格铁路：看清 for 的执行顺序，控制起点、边界和步长，用取模安排节拍，用累加器做出可靠的材料清单。

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 复制十九次，还把编号数错了



村长要一条20格铁路。史蒂夫复制、粘贴、改编号，写到第17格，竟贴了两次“第14格”，还宣布这是双层保险。矿车不赞同，矿车只会咣当。

Alex 推来循环建造机：告诉它从哪里开始、什么时候停、每次走几格，重复任务便不必写二十遍。for 适合次数或数列清楚的工作；它和分支、变量合作，才会得到有用的施工清单。

- for 圆括号里的三段，分别在什么时候执行？
- < 与 <= 会让哪一格消失或多出来？
- 计数器与累加器分别回答什么问题？

本课打印控制台施工清单。动力铁轨间隔、火把间隔、材料单价均为练习规则，不是游戏合成配方，也不会放置真实方块。

02 for 的三格机关与一条回路

初始化只在进入循环时执行一次；每轮开始前检查条件；条件为true才做循环体；本轮结束后更新，再回去检查。不是反复从左到右扫描三个格子。

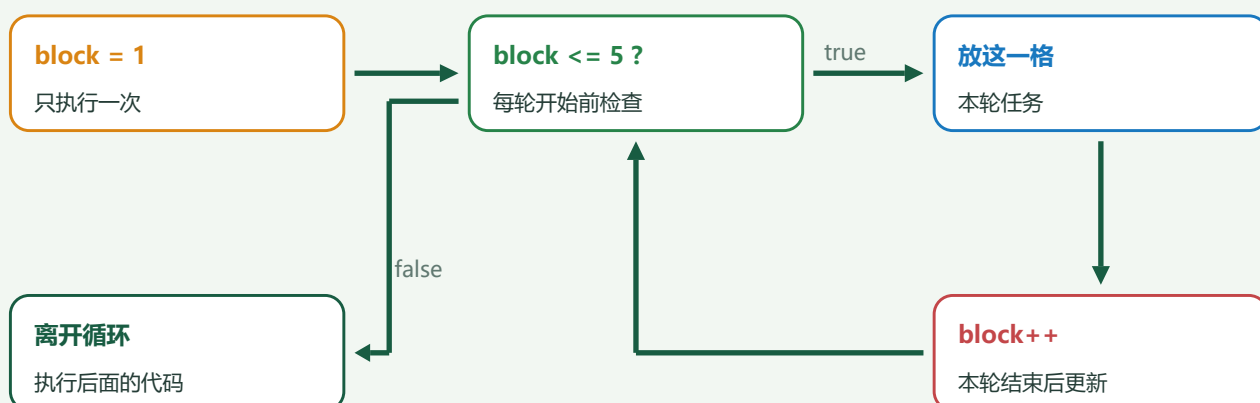
片段：放在 main 中，单独运行

```
for (int block = 1; block <= 5; block++) {  
    System.out.println("第" + block + "格放金砖");  
}
```

运行核对

第1格放金砖
第2格放金砖
第3格放金砖
第4格放金砖
第5格放金砖

顺着箭头走：条件为false才离开



最后一轮使用 block=5。更新后变成6，再检查 6 <= 5 得到false，于是停止。block 在 for 的初始化处声明，只在这个 for 内有效。

本课短例子分别放在熟悉的 class 与 main 外壳内运行。先写数列，再按运行；完整 RailwayPlanner 程序在后面两页给出。

03 起点、边界与步长

写出经过的数列，再数循环次数

循环设置	经过的数值	次数
0开始, < 4, +1	0、1、2、3	4
1开始, <= 4, +1	1、2、3、4	4
2开始, <= 10, +2	2、4、6、8、10	5
2开始, <= 9, +2	2、4、6、8	4

片段：放在 main 中，单独运行

```
for (int slot = 0; slot < 4; slot++) {
    System.out.print(slot + " ");
}
System.out.println();
```

运行核对

0 1 2 3

片段：放在 main 中，单独运行

```
for (int position = 2; position <= 10; position += 2) {
    System.out.print(position + " ");
}
System.out.println();
```

运行核对

2 4 6 8 10

i 是常见变量名，position、layer、slot 往往更能说明任务。i++ 每次加1，i += 2 每次加2。步长可能跨过边界，不保证一定恰好落在终点。

纸上追踪至少写三列：本轮值、条件真假、输出。循环次数取决于起点、条件和更新三者，不能只看终点数字。

04 倒计时：更新要向终点靠近

片段：放在 main 中，单独运行

```
for (int second = 5; second >= 1; second--) {
    System.out.println(second + "...");
}
System.out.println("出发! ");
```

运行核对

5...
4...
3...
2...
1...
出发!

倒数数列：包含1，不包含0

5、4

先检查，再输出当前秒数。



3、2

每轮结束都减1。



1

输出后减到0，下次检查失败。

片段：放在 main 中，单独运行

```
for (int y = 10; y >= 2; y -= 2) {
    System.out.print(y + " ");
}
System.out.println();
```

运行核对

10 8 6 4 2

把 `>= 1` 改成 `> 1`，1就消失。若把 `second--` 错写成 `second++`，数字会朝错误方向走，短时间内刷出大量输出。立即停止，再检查更新方向。

不要用“等它自己停”排查错误循环。int 还可能在极端值处溢出，因此“方向写错就永远true”也不是严谨的解释。本课只使用明确、有限的小范围。

05 取模：给铁路打节拍

`position % 5 == 0` 表示位置能被5整除：5、10、15、20会触发。循环负责访问每一格，if 负责决定这一格要做什么。

两种节拍可以在同一格相遇

位置	<code>% 5 == 0</code>	<code>% 4 == 0</code>
4	false: 普通铁轨	true: 加火把
5	true: 动力铁轨	false: 不加火把
10	true: 动力铁轨	false: 不加火把
20	true: 动力铁轨	true: 加火把

片段：放在 main 中，单独运行

```
for (int position = 1; position <= 20; position++) {  
    if (position % 5 == 0) {  
        System.out.println(position + ": 动力铁轨");  
    }  
}
```

运行核对

5: 动力铁轨
10: 动力铁轨
15: 动力铁轨
20: 动力铁轨

片段：放在 main 中，单独运行

```
for (int lamp = 1; lamp <= 4; lamp++) {  
    String color = lamp % 2 == 0 ? "蓝灯" : "黄灯";  
    System.out.println(lamp + ": " + color);  
}
```

运行核对

1: 黄灯
2: 蓝灯
3: 黄灯
4: 蓝灯

条件？真时的值：假时的值，是简短的二选一表达式。第20格的铁轨与火把要独立判断，别用 `else if` 把第二个事件吞掉。

06 累加器：别把旧箱子扔掉

计数器记“第几轮”，累加器记“目前一共多少”。累加器在循环外初始化一次，再在循环内加入本轮收获，结束后才能拿到总量。

片段：放在 main 中，单独运行

```
int totalRails = 0;
for (int trip = 1; trip <= 5; trip++) {
    totalRails += 2;
    System.out.println("第" + trip + "趟, 共" + totalRails + "根");
}
```

运行核对

第1趟, 共2根
第2趟, 共4根
第3趟, 共6根
第4趟, 共8根
第5趟, 共10根

每轮加入新收获，旧值仍保留

本轮 ore	进入时 total	加入数量	更新后 total
1	0	1	1
2	1	2	3
3	3	3	6
4	6	4	10
5	10	5	15

片段：放在 main 中，单独运行

```
int total = 0;
for (int ore = 1; ore <= 5; ore++) {
    total += ore;
}
System.out.println(total);
```

运行核对

15

07 循环急诊室：四问诊断法



先查数列，再查保留状态

- 起点是什么？先判断就不成立，允许一轮都不执行。
- 最后一个合法值要不要包含？需要20就不能停在19。
- 更新后更接近哪一扇出口？步长或方向写错会改变数列。
- 哪些值要留到下一轮？累加器不要每轮重新清零。

可运行反例：每轮清零，算不出总和6

```
for (int i = 1; i <= 3; i++) {  
    int total = 0; // 反例：每轮新建箱子  
    total += i;  
    System.out.println(total);  
}
```

运行核对

```
1  
2  
3
```

循环体内的 total 每轮重新建立，块外也无法访问。想得到总和，把声明移到循环外，最后再输出一次。

怀疑循环跑飞时先停止程序，随后只改一个原因。不要为了看看坏在哪里，故意运行会无限刷屏的反例。

08 先写数列，再核对输出

让循环留下脚印

数列

列出计数器依次经过的值。



状态

写出累计量每轮的变化。



停止

解释最后一次条件为何为false。

片段：放在 main 中，单独运行

```
for (int block = 1; block <= 10; block += 3) {  
    System.out.print(block + " ");  
}  
System.out.println();
```

片段：放在 main 中，单独运行

```
for (int depth = 5; depth < 5; depth++) {  
    System.out.println("挖掘");  
}  
System.out.println("结束");
```

片段：放在 main 中，单独运行

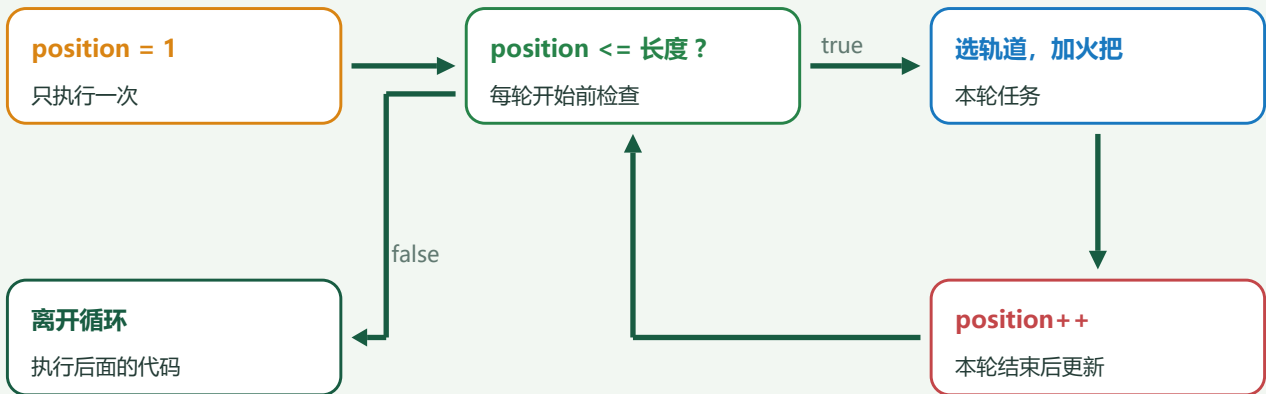
```
int powered = 0;  
for (int p = 1; p <= 12; p++) {  
    if (p % 4 == 0) {  
        powered++;  
    }  
}  
System.out.println(powered);
```

核对：第一题数列1、4、7、10，下一步13超界；第二题只输出“结束”，第一轮前条件已为false；第三题输出3，在4、8、12处计数。

09 项目蓝图：铁路施工清单

施工清单要打印位置、轨道类型和附加火把，最后汇总材料。长度用常量设置，允许0到60；0代表空计划，负数与超长计划报告错误。

每格施工一次，更新后再检查



练习施工契约

位置条件	施工内容	统计动作
所有合法位置	打印两位编号	本轮只打印一行
能被5整除	动力铁轨	数量+1, 预算+6
其他位置	普通铁轨	预算+1
能被4整除	附加火把	火把数量+1

单价是本课自定的铁锭预算模型，不是实际合成配方。20格中动力铁轨4格、普通铁轨16格，因此练习预算为 $4 \times 6 + 16 \times 1 = 40$ 。

10 施工代码：选轨道与记预算

新建 RailwayPlanner.java。完整程序分两页印刷，按顺序放入同一文件，不要在两页之间补花括号。

RailwayPlanner.java • 上半部分，下一页继续

```
public class RailwayPlanner {
    public static void main(String[] args) {
        final int SEGMENTS = 20;
        int poweredRails = 0, torches = 0, ironCost = 0;
        if (SEGMENTS < 0 || SEGMENTS > 60) {
            System.out.println("长度必须在0到60之间");
        } else {
            System.out.println("=== 铁路施工清单 ===");
            for (int position = 1; position <= SEGMENTS; position++) {
                boolean powered = position % 5 == 0;
                boolean torch = position % 4 == 0;
                String rail;
                if (powered) {
                    rail = "动力铁轨";
                    poweredRails++;
                    ironCost += 6;
                } else {
                    rail = "普通铁轨";
                    ironCost += 1;
                }
            }
        }
    }
}
```



position 是计数器，poweredRails、torches、ironCost 是三只材料箱。final 表示这里的长度设定不再重新赋值，换测试长度时修改初始化数字。

11 施工代码：加火把与统一汇总

RailwayPlanner.java • 紧接上一页

```

        if (torch) {
            torches++;
        }
        String extra = torch ? " + 火把" : "";
        System.out.printf("[%02d] %s%s%n", position, rail, extra);
    }
    System.out.println("--- 练习材料汇总 ---");
    System.out.println("铁路格数: " + SEGMENTS);
    System.out.println("动力铁轨: " + poweredRails);
    System.out.println("火把: " + torches);
    System.out.println("铁锭预算: " + ironCost);
}
}
}

```

20格标准计划，先手算再核对

动力铁轨

5、10、15、20，共4格。

火把

4、8、12、16、20，共5个。

铁锭预算

$16 \times 1 + 4 \times 6$ ，合计40。

%02d 把1显示为01；两个 %s 放入轨道名与附加文字；%n 使用平台换行。火把的 if 独立于轨道选择，第20格才能同时出现两种事件。

关键行应为 [04] 普通铁轨 + 火把、[05] 动力铁轨、[20] 动力铁轨 + 火把。循环结束后再打印汇总，避免把中间结果当最终预算。

12 验收：末格不能悄悄消失

逐次修改 SEGMENTS，记下真实结果

长度	施工行数	动力铁轨	火把	预算
0	0	0	0	0
1	1	0	0	1
20	20	4	5	40
21	21	4	5	41
60	60	12	15	120
-1或61	报范围错误	不施工	不统计	不统计

改坏一次，学会解释一次

- 把 `<= SEGMENTS` 改成 `< SEGMENTS`：长度20时只做到19，预算少6。
- 把 `position++` 改成 `position += 2`：只访问奇数格，20不会出现。
- 把累计量放进循环每轮清零：比较本轮数字与总量。
- 恢复标准版20，核对首格、末格、20条施工行和4、5、40三个汇总。

提交至少4组“设定、预测、实际、解释”。一次只改一个地方，才能知道哪个规则影响了结果。

13 再造一段自己的方块世界



木镐任务：三十格城墙

输出1到30格，第10、20、30格改为加固柱。核对首尾与30条记录；终点改成0时，不能施工任何一格。

铁镐任务：一百颗经验球

累计1到100的总和，同时统计7的倍数。总和应为5050，倍数共14个；不能把答案直接写进输出，先画前5轮 total 的变化。

钻石镐任务：输入长度的城墙

读取1到60的长度，每6格加火把，每10格加瞭望台，分别统计。30格处两种装置都要出现。测试1、6、10、30、60、0、61和 abc。

复盘：初始化一次，条件每轮先查，循环体完成后更新。数列由起点、边界与步长一起决定，累加器在循环外准备。下一课将用 while 处理次数未知的搜索。

技术核对：Oracle Java 语言规范第14章，基本 for 与代码块。

<https://docs.oracle.com/javase/specs/jls/se21/html/jls-14.html>