第11课

用循环画像素建筑

嵌套循环、二维坐标与一座像素房子



一行代码让施工队动起来，一组坐标让每个方块找到自己的位置。你将完成严格9列、7行的房子，并解释门窗为什么不会被墙盖住。
前置：变量、条件、for 与 while。建议先完成第9、10课。

阅读方式：先猜结果，再运行核对；每次只改一个条件。

本章使用 JDK 21，无需预览功能。故事与插画借用方块世界，练习运行在普通 Java 控制台。

01 两位建筑师，先约好顺序



村长要一座宽9格、高7格的房子。阿行负责换楼层，小列负责在当前楼层从左走到右。如果两人各走各的，房子只会变成一团石砖毛线。

程序里的约定很清楚：外层循环选中一行，内层循环走完这一行的所有列，然后换行。屏幕的行号向下增加，列号向右增加；这与“往上盖楼”的故事方向不同，要以坐标图为准。

- row 决定当前行，col 决定当前列。
- 每走完一行，内层循环会重新从 col = 0 开始。
- 每个格子打印一个字符；println 只负责结束整行。

02 两层循环怎样配合

先选行，再把本行的列走完

外层 row
选当前行

→

内层 col
从0扫到最后一列

→

换行
本行完成，再选下一行

L11Example02.java • 完整独立示例

```
public class L11Example02 {
    public static void main(String[] args) {
        for (int row = 0; row < 2; row++) {
            for (int col = 0; col < 3; col++) {
                System.out.print(row + "," + col + " ");
            }
            System.out.println();
        }
    }
}
```

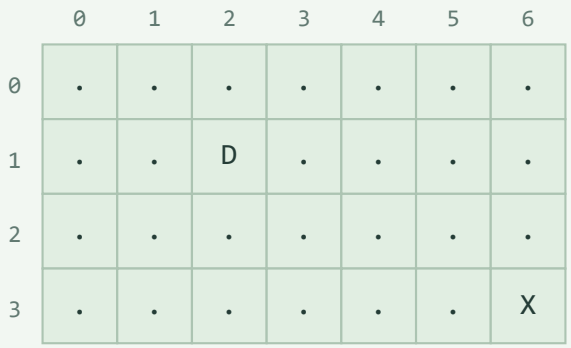
运行核对

```
0,0 0,1 0,2
1,0 1,1 1,2
```

外层2轮，每轮内层3次，总共打印6组坐标。println 放在内层外面，表示“一行做完了”。

03 给每个格子一张地址牌

4行7列：右下角是 (3,6)



行列都从0开始。4行的合法行号是0到3，7列的合法列号是0到6。图中 D 的地址是 (1,2)，X 是右下角 (3,6)。

L11Example03.java • 完整独立示例

```
public class L11Example03 {
    public static void main(String[] args) {
        for (int row = 0; row < 4; row++) {
            for (int col = 0; col < 7; col++) {
                System.out.print(row == 1 && col == 2 ? "D" : ".");
            }
            System.out.println();
        }
    }
}
```

运行核对

```
.......
..D....
.......
.......
```

把“我要在这里放门”翻译成条件：row == 1 && col == 2。位置只是一对整数，并没有藏着魔法。

04 四条边界，一圈围墙

4行7列的空心边框

	0	1	2	3	4	5	6
0	#	#	#	#	#	#	#
1	#						#
2	#						#
3	#	#	#	#	#	#	#

L11Example04.java • 完整独立示例

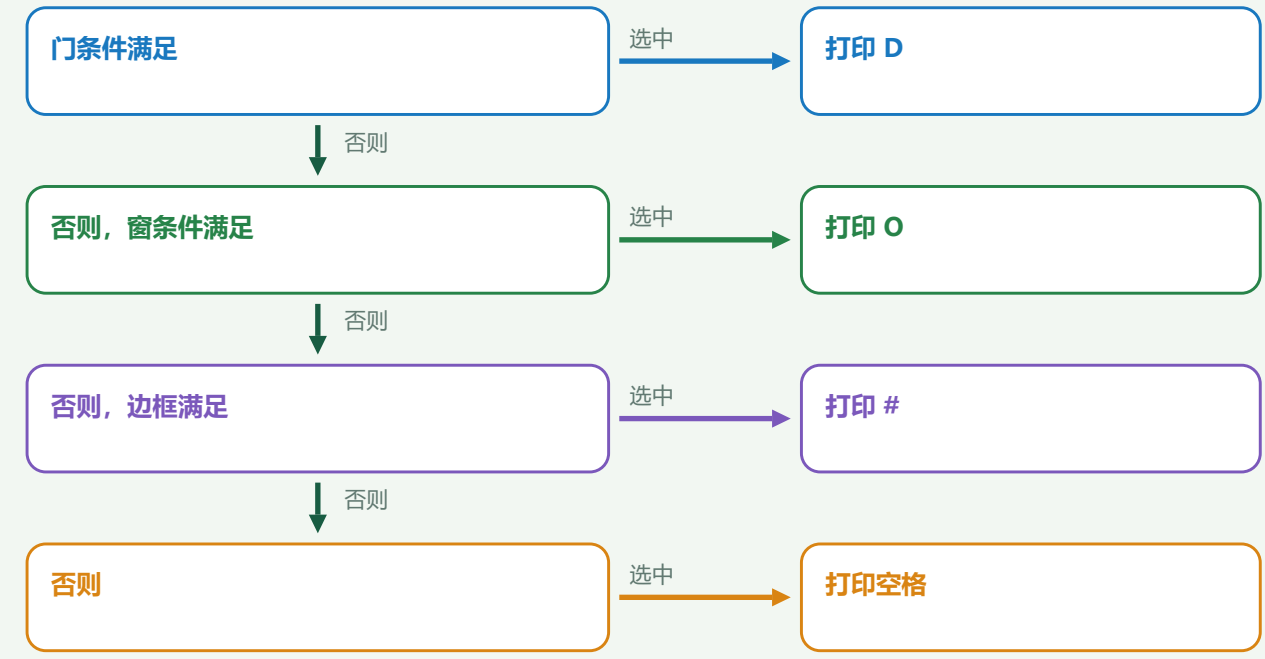
```
public class L11Example04 {
    public static void main(String[] args) {
        int rows = 4, cols = 7;
        for (int row = 0; row < rows; row++) {
            for (int col = 0; col < cols; col++) {
                boolean edge = row == 0 || row == rows - 1
                        || col == 0 || col == cols - 1;
                System.out.print(edge ? "#" : " ");
            }
            System.out.println();
        }
    }
}
```

运行核对

```
#######
#     #
#     #
#######
```

逻辑或表示四条边有任意一条命中即可。角落满足两条边，但程序每格只打印一次，因此不会重复。共有18个边框字符。

05 让门窗先占位

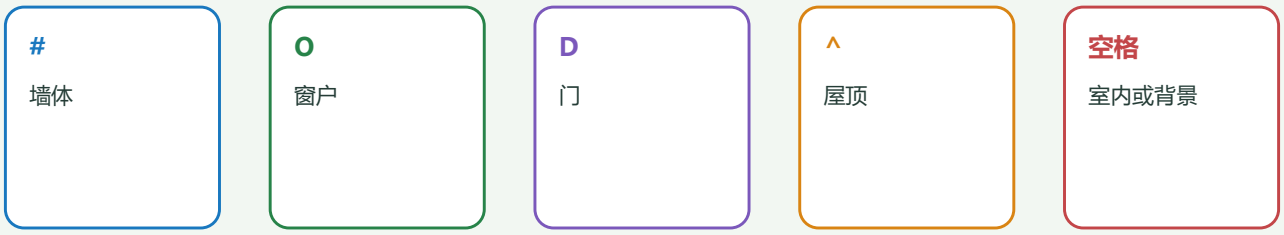


门可能就在底边，窗可能覆盖墙。若先检查墙，墙会抢走这些格子。使用 if / else if / else，让一格只选择一种符号；把更具体的门窗条件放在墙条件前面。

条件之间不是同时打印四层图案，而是从上到下找到第一个满足的分支。为门窗增加坐标测试，比盯着房子猜更可靠。

06 符号也需要一张材料表

每种符号只负责一种材料



像素画靠字符宽度维持队形。选等宽字体和半角符号；中文、表情与部分 Unicode 符号的显示宽度可能不同。每格只用一个 ASCII 字符，便于精确验收。

L11Example06.java • 完整独立示例

```
public class L11Example06 {
    public static void main(String[] args) {
        for (int row = 0; row < 4; row++) {
            for (int col = 0; col < 6; col++) {
                System.out.print((row + col) % 2 == 0 ? "#" : ".");
            }
            System.out.println();
        }
    }
}
```

运行核对

```
#.#.#.
.#.#.#
#.#.#.
.#.#.#
```

奇偶相间的秘密是 row + col。向右或向下走一格，和都会改变1，因此颜色轮换。

07 建筑故障诊所



故障实验：for 中声明的 row 不能在循环范围外使用。下面故意保留错误，请先预测，再尝试修复。

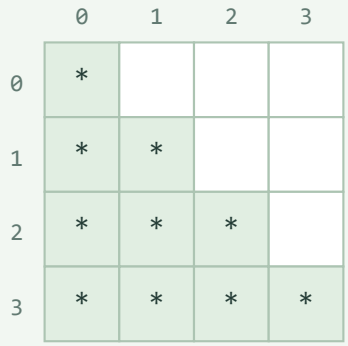
BadRow.java • 故意错误示例

```
public class BadRow {
    public static void main(String[] args) {
        for (int row = 0; row < 2; row++) { }
        System.out.println(row);
    }
}
```

- 图案变成一长列：检查 println 是否被放进内层循环。
- 少了最后一列：检查 col < cols 的边界，不能提前停在 cols - 1。
- 永远画不出右边墙：合法最后列是 cols - 1，不是 cols。

08 先预测，再允许电脑发言

4行小三角：每行多一颗星



L11Example08.java • 完整独立示例

```
public class L11Example08 {
    public static void main(String[] args) {
        for (int row = 0; row < 4; row++) {
            for (int col = 0; col <= row; col++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

运行核对

```
*
**
***
****
```

本例没有固定打印4列，而是 row + 1 个星号。先区分“每行都扫完整宽度”和“每行扫不同长度”，再判断边界是否正确。

09 挑战蓝图：严格9列7行

像素房子的完整坐标图

	0	1	2	3	4	5	6	7	8
0			^	^	^	^	^		
1		^	^	^	^	^	^	^	
2	#	#	#	#	#	#	#	#	#
3	#	O			D			O	#
4	#	O			D			O	#
5	#				D				#
6	#	#	#	#	#	#	#	#	#

先画图，再写条件。两行屋顶分别占第0行的2到6列、第1行的1到7列；门位于第4列、第3到5行；窗位于第1或7列、第3或4行。第2行是上墙，第6行是地基，两侧墙在第0和8列。

- 房子每行必须包含9个字符，前后空格也算。
- 判断顺序：屋顶、门、窗、墙、空格。
- 背景空格仍占一个格子，不能直接跳过打印。

10 完整实现：一格一个决定



PixelHouse.java • 完整独立示例

```
public class PixelHouse {
    public static void main(String[] args) {
        int rows = 7, cols = 9;
        for (int row = 0; row < rows; row++) {
            for (int col = 0; col < cols; col++) {
                char block;
                boolean roof = row == 0 && col >= 2 && col <= 6
                        || row == 1 && col >= 1 && col <= 7;
                boolean door = col == 4 && row >= 3 && row <= 5;
                boolean window = (col == 1 || col == 7)
                        && (row == 3 || row == 4);
                boolean wall = row >= 2 && (row == 2 || row == 6
                        || col == 0 || col == cols - 1);
                if (roof) block = '^';
                else if (door) block = 'D';
                else if (window) block = 'O';
                else if (wall) block = '#';
                else block = ' ';
                System.out.print(block);
            }
            System.out.println();
        }
    }
}
```

运行核对

```
  ^^^^^
 ^^^^^^^
#########
#O  D  O#
#O  D  O#
#   D   #
#########
```

所有代码段按顺序接在 PixelHouse.java 中。屋顶两行保留左右空格；统计字符数量时，空格也要算。

11 用边界测试替房子验收

三个尺寸检查，三个常见原因

7行
外层执行7轮

9列
内层执行9次

门窗可见
具体条件先于墙

验收时数行、数列，再检查关键地址：(0,2)是屋顶，(3,4)是门，(3,1)是窗，(6,8)是墙。只说“看起来像房子”还不够。

- 把 col < cols 改成 col <= cols：每行多一格，找到原因后恢复。
- 把 println 移进内层：解释为什么每个方块独占一行，再恢复。
- 增加火把：在墙条件前检查 row == 2 && col == 4，该行应变为 ####T####。
- 把通用边框例的 rows 改为1：应只输出一行；这里不要直接改固定房子的坐标。

测试结束恢复原程序，再核对严格9列7行。每次只改一项，否则你很难判断是哪块砖把墙推歪了。

12 轮到你设计建筑



- 木镐：画6×6空心边框，测试1行、2行及正常尺寸。
- 铁镐：画8×8棋盘，再改成9列；解释奇偶条件。
- 钻石镐：设计9×9苦力怕脸，先交坐标草图，再写眼睛与嘴的条件。

交付源代码、运行图案、行列统计和一条 Bug 记录。能解释一个坐标为什么命中某种符号，比背下整段代码更有用。

下一课：把重复动作装进方法。先保持房子输出不变，再把屋顶和墙体拆成能读懂名字的步骤。