

JAVA / LESSON 11 OF 40

Build Pixel Art with Loops

Nested loops, 2D coordinates and a pixel house

This chapter uses JDK 21 and does not require the preview function. | 90-110 minutes



Make a pixel house with 9 columns and 7 rows

A line of code makes the construction team move, and a set of coordinates allows each block to find its position. You will complete a house with strictly 9 columns and 7 rows and explain why the doors and windows are not covered by the walls.

Preposition: variables, conditions, for and while. It is recommended to complete Lessons 9 and 10 first.

Before you start: It is recommended to complete Lessons 9-10 first

Read the full lesson and copy the code:

<https://teacher-ma.de/minecraft-java-lesson.php?lesson=11&lang=en>

Long code lines may wrap at the > margin marker. The marker is not part of the code.

In this lesson

01. Two architects, make an appointment first
02. How to cooperate with the two layers of circulation
03. Give each grid an address card
04. four borders, a circle of walls
05. Let the doors and windows take up space first
06. symbols also require a material list
07. Building Failure Clinic
08. Predict first, then allow the computer to speak
09. Challenge Blueprint: Strictly 9 columns and 7 rows
10. Complete implementation: one decision per grid
11. Use boundary testing for house acceptance
12. It's your turn to design the building

01 / 12

Two architects, make an appointment first



The village chief wants a house that is 9 blocks wide and 7 blocks high. Ah Xing is responsible for changing floors, and Xiao Li is responsible for walking from left to right on the current floor. If the two go their separate ways, the house will become nothing more than a ball of stone bricks and wool.

The agreement in the program is very clear: the outer loop selects a row, the inner loop completes all the columns of this row, and then changes to a new line. The row numbers of the screen increase downward and the column numbers increase to the right; this is different from the direction of the story of "building a building upwards" and should be based on the coordinate diagram.

- row determines the current row, col determines the current column.
- Every time a row is completed, the inner loop will start again from col = 0.
- Each grid prints one character; println is only responsible for ending the entire line.

02 / 12

How to cooperate with the two layers of circulation

Select a row first, then complete the columns of this row

Step	Explanation
1. Outer row	Select current row
2. Inner layer col	Scan from 0 to the last column
3. Line feed	This row is completed, then select the next row

L11Example02.java · Complete independent example

```
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();  
        }  
    }  
}
```

Output

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

2 rounds of outer layer, 3 times of inner layer in each round, printing 6 sets of coordinates in total. println is placed outside the inner layer to mean "one line is finished".

03 / 12

Give each grid an address card

4 rows and 7 columns: the lower right corner is (3,6)

row / col	0	1	2	3	4	5	6
0	.	.	.	.	.	.	.
1	.	.	D	.	.	.	.
2	.	.	.	.	.	.	.
3	.	.	.	.	.	.	X

The rows and columns all start from 0. The legal row numbers for 4 rows are 0 to 3, and the legal column numbers for 7 columns are 0 to 6. The address of D in the figure is (1,2), and X is the lower right corner (3,6).

L11Example03.java · Complete independent example

```
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();
        }
    }
}
```

Output

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

Translate "I want to put a door here" into conditions: `row == 1 && col == 2`. The position is just a pair of integers, there is no magic hidden there.

four borders, a circle of walls

Hollow border of 4 rows and 7 columns

row / col	0	1	2	3	4	5	6
0	#	#	#	#	#	#	#
1	#						#
2	#						#
3	#	#	#	#	#	#	#

L11Example04.java · Complete independent example

```
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();  
        }  
    }  
}
```

Output

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

Logical OR means that any one of the four edges can be hit. The corners satisfy two sides, but the program only prints each grid once, so it will not be repeated. There are 18 border characters in total.

05 / 12

Let the doors and windows take up space first

There are multiple possibilities for the same grid, choose once according to priority

Check in order	On a match
The gate condition is satisfied	Print D
Otherwise, the window condition satisfies	Print O
Otherwise, the border satisfies	Print #
Otherwise	Print spaces

Check from top to bottom; execute only the first matching branch.

The door may be on the bottom edge, and the window may cover the wall. If you check the wall first, the wall will take away these cells. Use if / else if / else to select only one symbol per cell; put more specific door and window conditions before the wall conditions.

Instead of printing four layers of patterns simultaneously among the conditions, the first satisfying branch is found from top to bottom. Adding coordinate testing for doors and windows is more reliable than guessing by staring at the house.

06 / 12

symbols also require a material list

Each symbol is only responsible for one material

Element	Explanation
#	Wall
O	Windows
D	The door.
^	Roof
Slot	Interior or background

Pixel art relies on character width to maintain formation. Select a fixed-width font and half-width symbols; the display width of Chinese, emoticons, and some Unicode symbols may be different. Use only one ASCII character per cell to facilitate accurate acceptance.

L11Example06.java · Complete independent example

```
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();  
        }  
    }  
}
```

Output

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

The secret to odd and even is $row + col$. Move one space to the right or down, and both the and will change by 1, so the colors rotate.

07 / 12

Building Failure Clinic



Fault experiment: row declared in for cannot be used outside the loop scope. Errors are intentionally left below, please anticipate them first and then try to fix them.

BadRow.java · Intentional error example

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

- The pattern becomes a long column: check if println is put into the inner loop.
- The last column is missing: check the boundary of col < cols and cannot stop at cols - 1 early.
- The right wall can never be drawn: the legal last column is cols - 1, not cols.

08 / 12

Predict first, then allow the computer to speak

4 rows of small triangles: one more star in each row

row / col	0	1	2	3
0	*			
1	*	*		
2	*	*	*	
3	*	*	*	*

L11Example08.java · Complete independent example

```
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();
        }
    }
}
```

Output

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

This example does not print 4 fixed columns, but row + 1 asterisk. First distinguish between "scan the full width of each line" and "scan each line of different lengths", and then determine whether the boundary is correct.

09 / 12

Challenge Blueprint: Strictly 9 columns and 7 rows

Complete coordinate map of pixel house

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

Draw the picture first, then write the conditions. The two rows of roofs occupy columns 2 to 6 of row 0 and columns 1 to 7 of row 1 respectively; doors are located in columns 4 and rows 3 to 5; windows are located in columns 1 or 7 and rows 3 or 4. Row 2 is the upper wall, row 6 is the foundation, and the two side walls are in columns 0 and 8.

- Each line of the house must contain 9 characters, including spaces before and after.
- Judgment order: roof, door, window, wall, space.
- The background space still occupies one grid, and printing cannot be skipped directly.

10 / 12

Complete implementation: one decision per grid



PixelHouse.java · Complete standalone example

```
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();  
        }  
    }  
}
```

Output

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

All code segments are connected in order in PixelHouse.java. The two lines on the roof retain left and right spaces; when counting the number of characters, spaces are also counted.

11 / 12

Use boundary testing for house acceptance

Three size checks, three common reasons

Element	Explanation
7 lines	The outer layer executes 7 rounds
9 columns	The inner layer is executed 9 times
Doors and windows visible	Specific conditions precede the wall

When accepting, count the rows and columns, and then check the key addresses: (0,2) is the roof, (3,4) is the door, (3,1) is the window, and (6,8) is the wall. It's not enough to say it "looks like a house."

- Change `col < cols` to `col <= cols`: one more space per line, and restore after finding the cause.
- Move `println` into the inner layer: explain why each square is on its own line, and then restore it.
- Add torch: Check `row == 2 && col == 4` before wall condition, the row should become `#####T#####`.
- Change the rows of the general border example to 1: only one row should be output; do not directly change the coordinates of the fixed house here.

After the test, restore the original program, and then check strictly 9 columns and 7 rows. Only change one item at a time, otherwise it will be difficult to tell which brick is pushing the wall crooked.

12 / 12

It's your turn to design the building



- Wooden pick: draw 6×6 hollow border, test 1 row, 2 rows and normal size.
- Iron pick: draw an 8×8 chessboard, and then change it to 9 columns; explain the odd and even conditions.
- Diamond Pickaxe: To design a 9×9 Creeper face, first submit a coordinate sketch, and then write the conditions for the eyes and mouth.

Deliver source code, running patterns, row and column statistics and a bug record. Being able to explain why a coordinate hits a certain symbol is more useful than memorizing the entire code.

Next lesson: Put repetitive actions into methods. First keep the output of the house unchanged, and then dismantle the roof and walls into steps where the names can be read.