

JAVA / LESSON 09 OF 40

The Loop Builder

Let the counters keep track, let the cumulators get harvest.

Java 11+ | 90-110 minutes



Make railway construction list

From five grids of stone bricks to twenty grids of railways: see clearly the execution sequence of for, control the starting point, boundary and step size, use modulo to arrange the beat, and use the accumulator to make a reliable material list.

Before you start: It is recommended to complete Lessons 1-8 first

Read the full lesson and copy the code:

<https://teacher-ma.de/minecraft-java-lesson.php?lesson=9&lang=en>

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

In this lesson

01. Copied nineteen times and counted the wrong number
02. for's three-grid mechanism and one circuit
03. starting point, boundary and step length
04. Countdown: Update is approaching the end point
05. Taking the mold: Beating the railway
06. Accumulator: Don't throw away old boxes
07. Circulation Emergency Room: Four Questions Diagnosis Method
08. Write the sequence first, then check the output
09. Project Blueprint: Railway Construction Checklist
10. Construction code: track selection and budget recording
11. Construction Code: Adding Torches and Unifying Summary
12. Acceptance: The last square cannot disappear quietly
13. Recreate a block world of your own

01 / 13

Copied nineteen times and counted the wrong number



The village chief wants a 20-block railway. Steve copied, pasted, and adapted the number until he reached the 17th box. He posted "14th box" twice and announced that this was double insurance. The minecart doesn't agree, the minecart will just clang.

Alex introduced the loop builder: tell it where to start, when to stop, and how many squares to move at a time, so you don't have to write twenty times to repeat the task. `for` is suitable for work with clear times or sequences; it works with branches and variables to get a useful construction list.

- When will the three paragraphs in `for` parentheses be executed?
- Which box will disappear or appear added by `<` and `<=`?
- What questions do counters and accumulators answer respectively?

This lesson prints the console construction list. The power rail spacing, torch spacing, and material unit prices are all practice rules, not game synthesis recipes, and real blocks will not be placed.

02 / 13

for's three-grid mechanism and one circuit

Initialization is only executed once when entering the loop; the condition is checked before the start of each round; the loop body is made only when the condition is true; it is updated after the end of the current round and goes back to check. Instead of scanning three grids from left to right repeatedly.

Snippet: place inside main and run separately

```
for (int block = 1; block <= 5; block++) {  
    System.out.println("No." + block + "Placing stone bricks");  
}
```

Output

```
No.1Placing stone bricks  
No.2Placing stone bricks  
No.3Placing stone bricks  
No.4Placing stone bricks  
No.5Placing stone bricks
```

Follow the arrow: Leave only when the condition is false

Step	Action
1 · Once	block = 1
2 · Check	block <= 5 ?
3 · If true	Put this box
4 · Update; return to step 2	block++

Exit the loop when the condition is false.

The final round uses block=5. After updating, it becomes 6, and then checks 6 <= 5 and gets false, so it stops. Block is declared at the initialization point of for and is only valid within this for.

The short examples in this lesson are run in the familiar class and main shells. Write the sequence first and then press Run; the complete RailwayPlanner program is given on the next two pages.

03 / 13

starting point, boundary and step length

Write out the passed sequence, and then count the number of cycles

Cycle settings	Passed value	times
Starting from 0, < 4, +1	0、 1、 2、 3	4
1 starts, <= 4, +1	1、 2、 3、 4	4
2 starts, <= 10, +2	2、 4、 6、 8、 10	5
2 starts, <= 9, +2	2、 4、 6、 8	4

Snippet: place inside main and run separately

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

Output

```
0 1 2 3
```

Snippet: place inside main and run separately

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

Output

```
2 4 6 8 10
```

`i` is a common variable name, and `position`, `layer`, and `slot` are often more descriptive of the task. `i++` adds 1 each time, `i += 2` adds 2 each time. The step length may cross the boundary and is not guaranteed to fall exactly at the end point.

Write at least three columns for tracking on paper: current rotation, true and false conditions, and output. The number of loops depends on the starting point, conditions and updates. You cannot just look at the end number.

04 / 13

Countdown: Update is approaching the end point

Snippet: place inside main and run separately

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

Output

```
5...  
4...  
3...  
2...  
1...  
Departure!
```

Reciprocal sequence: contains 1, does not contain 0

Step	Explanation
1. 5, 4	Check first, then output the current number of seconds.
2. 3, 2	Decrease 1 at the end of each round.
3. 1	After output, it is reduced to 0 and the next check fails.

Snippet: place inside main and run separately

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

Output

```
10 8 6 4 2
```

Change `>= 1` to `> 1`, and 1 will disappear. If `second--` is mistakenly written as `second++`, the numbers will go in the wrong direction and a large amount of output will be generated in a short period of time. Stop immediately and check the update direction again.

Do not use "wait for it to stop" to troubleshoot error loops. Int may also overflow at extreme values, so "if the direction is written wrong, it will always be true" is not a rigorous explanation. This lesson uses only a clear, limited scope.

05 / 13

Taking the mold: Beating the railway

`position % 5 == 0` means the position is divisible by 5: 5, 10, 15, 20 will trigger. The loop is responsible for visiting each cell, and the if is responsible for deciding what to do with this cell.

Two beats can meet in the same grid

Location	% 5 == 0	% 4 == 0
4	false: Ordinary rails	true: add torch
5	true: powered rail	false: No torch added
10	true: powered rail	false: No torch added
20	true: powered rail	true: add torch

Snippet: place inside main and run separately

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

Output

```
5: Power rail  
10: Power rail  
15: Power rail  
20: Power rail
```

Snippet: place inside main and run separately

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

Output

```
1: Yellow light
2: Blue light
3: Yellow light
4: Blue light
```

Condition ? Value when true: Value when false, is a short alternative expression. The rails and torches in the 20th grid must be judged independently. Do not use else if to swallow up the second event.

06 / 13

Accumulator: Don't throw away old boxes

The counter records "which round", and the accumulator records "the current total number". The accumulator is initialized once outside the loop, and then the current round of harvest is added inside the loop, and the total amount can be obtained only after the end.

Snippet: place inside main and run separately

```
int totalRails = 0;
for (int trip = 1; trip <= 5; trip++) {
    totalRails += 2;
    System.out.println("No." + trip + "Trips, total" + totalRails + "root");
}
```

Output

```
No.1Trips, total2root
No.2Trips, total4root
No.3Trips, total6root
No.4Trips, total8root
No.5Trips, total10root
```

New harvest is added in each round, and the old value is still retained

This round ore	total when entering	Add quantity	Total after update
1	0	1	1
2	1	2	3
3	3	3	6
4	6	4	10
5	10	5	15

Snippet: place inside main and run separately

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

Output

```
15
```

Circulation Emergency Room: Four Questions Diagnosis Method



Check the sequence first, then check the retention status

- What is the starting point? If it is judged first, it will not be established, and it is allowed not to be executed in one round.
- Should the last legal value be included? If you need 20, you can't stop at 19.
- Which exit is closer after the update? Writing the wrong step size or direction will change the sequence.
- Which values should be saved for the next round? The accumulator should not be reset every round.

Runnable counterexample: Clear every round, cannot calculate the sum 6

```
for (int i = 1; i <= 3; i++) {  
    int total = 0; // Counterexample: Create a new box in each round  
    total += i;  
    System.out.println(total);  
}
```

Output

```
1  
2  
3
```

The total in the loop body is re-established every round and cannot be accessed outside the block. To get the sum, move the statement outside the loop and print it again at the end.

When it is suspected that the loop is running out, the program is stopped first, and then only one reason is changed. Don't deliberately run a counterexample that will refresh the screen infinitely just to see what's wrong.

08 / 13

Write the sequence first, then check the output

Let the cycle leave footprints

Step	Explanation
1. Sequence	List the values that the counter passes through in sequence.
2. Status	Write the change in the cumulative amount in each round.
3. Stop	Explain why the last condition is false.

Snippet: place inside main and run separately

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

Snippet: place inside main and run separately

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

Snippet: place inside main and run separately

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

Check: the first question number sequence 1, 4, 7, 10, the next step 13 is out of bounds; the second question only outputs "end", the condition before the first round is false; the third question outputs 3, counting at 4, 8, 12.

09 / 13

Project Blueprint: Railway Construction Checklist

The construction list should print the location, track type and additional torches, and finally summarize the materials. The length is set as a constant, allowing 0 to 60; 0 represents an empty plan, and negative numbers and overlong plans report an error.

Each grid is constructed once and then checked after updating.

Step	Action
1 · Once	position = 1
2 · Check	position <= length ?
3 · If true	Select track, add torch
4 · Update; return to step 2	position++

Exit the loop when the condition is false.

Practice Construction Contract

Location conditions	Construction content	Statistical actions
All legal locations	Print two-digit number	Only print one line this round
Divisible by 5	Power rail	Quantity + 1, budget + 6
Other locations	Ordinary Rail	Budget +1
Divisible by 4	Additional torch	Number of torches + 1

The unit price is the iron ingot budget model determined by this lesson, not the actual synthesis formula. Among the 20 squares, there are 4 squares for power rails and 16 squares for ordinary rails, so the practice budget is $4 \times 6 + 16 \times 1 = 40$.

10 / 13

Construction code: track selection and budget recording

Create new `RailwayPlanner.java`. The complete program is printed on two pages, placed in the same file in order, and do not add curly brackets between the two pages.

`RailwayPlanner.java` · First half, continue on next page

```
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("The length must be within 0 to 60 between");
        } else {
            System.out.println("=== Railway construction list ===");
            for (int position = 1; position <= SEGMENTS; position++) {
                boolean powered = position % 5 == 0;
                boolean torch = position % 4 == 0;
                String rail;
                if (powered) {
                    rail = "Power rail";
                    poweredRails++;
                    ironCost += 6;
                } else {
                    rail = "Ordinary Rail";
                    ironCost += 1;
                }
            }
        }
    }
}
```



`position` is a counter, `poweredRails`, `torches`, and `ironCost` are three material boxes. `final` means that the length setting here will not be reassigned, and the initialization number will be modified when changing the test length.

11 / 13

Construction Code: Adding Torches and Unifying Summary

RailwayPlanner.java · Next to the previous page

```
        if (torch) {
            torches++;
        }
        String extra = torch ? " + Torch" : "";
        System.out.printf("[%02d] %s%s%n", position, rail, extra);
    }
    System.out.println("--- Summary of practice materials ---");
    System.out.println("Railway grid number: " + SEGMENTS);
    System.out.println("Power rail: " + poweredRails);
    System.out.println("Torch: " + torches);
    System.out.println("Iron ingot budget: " + ironCost);
}
}
```

20 grid standard plan, first hand calculation and then check

Element	Explanation
Power rail	5, 10, 15, 20, a total of 4 grids.
Torch	4, 8, 12, 16, 20, 5 in total.
Iron ingot budget	16×1 + 4×6, total 40.

%02d displays 1 as 01; two %s put the track name and additional text; %n uses platform wrapping. The if of the torch is independent of the track selection, and only the 20th grid can have two events appear at the same time.

The key rows should be [04] Normal Rail + Torch, [05] Powered Rail, [20] Powered Rail + Torch. Print the summary after the loop ends to avoid using the intermediate results as the final budget.

12 / 13

Acceptance: The last square cannot disappear quietly

Modify SEGMENTS one by one and write down the real results

Length	Number of construction lines	Power rail	Torch	Budget
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 or 61	Report range error	No construction	No statistics	No statistics

Correct once, learn to explain once

- Change `<= SEGMENTS` to `< SEGMENTS`: when the length is 20, it can only be 19, and the budget is 6 less.
- Change `position++` to `position += 2`: Only odd cells are accessed, 20 will not appear.
- Put the accumulated amount into the cycle and clear it in each round: compare the current round number with the total amount.
- Restore the standard version 20, check the first grid, the last grid, the 20 construction lines and the three summaries of 4, 5, and 40.

Submit at least 4 groups of "setting, prediction, actual, explanation". Only by changing one place at a time can you know which rule affected the result.

13 / 13

Recreate a block world of your own



Wooden Pickaxe Task: Thirty Grid City Wall

Output 1 to 30 grids, and change the 10th, 20th, and 30th grids to reinforcement columns. Check the first and last records with 30 records; when the end point is changed to 0, no grid can be constructed.

Iron pickaxe task: one hundred experience balls

Accumulate the sum of 1 to 100, and count the multiples of 7 at the same time. The total should be 5050, and there are 14 multiples in total; the answer cannot be written directly into the output, so first draw the changes in total in the first five rounds.

Diamond Pickaxe Task: Enter the length of the city wall

Read the length from 1 to 60, add torches every 6 blocks, add observation towers every 10 blocks, and count them separately. Both devices will appear at 30 blocks. Test 1, 6, 10, 30, 60, 0, 61 and abc.

Review: initialize once, check the conditions first in each round, and update after the loop body is completed. The sequence is determined by the starting point, bounds, and step size, and the accumulator is prepared outside the loop. The next lesson will use while to handle an unknown number of searches.

Technical check: Chapter 14 of Oracle Java Language Specification, basic for and code blocks.

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