

Start, Stop, Step

for loops



Three lines run this loop

PAUSE AND TRY

Point to the three lines that control this loop.

FROM TOPIC 2.7

```
int seat = 1;
while (seat <= 4) {
    System.out.println("Seat " + seat);
    seat++;
}
```

Which of the three is easiest to forget? What happens if you forget it?

By the end of class you can...

- 1 Name the three parts of a for loop header and say what each one does.
- 2 Trace a for loop in the exact order Java runs it: initialization once, then test, body, update.
- 3 Change where a loop starts, where it stops, and how it steps, and count its iterations.
- 4 Spot off by one errors and a stray semicolon after the header.
- 5 Rewrite a for loop as a while loop, and a counting while loop as a for loop.

A large white square with rounded corners, tilted slightly to the right, containing a large, bold, black number '1'.

PART 1

One header, three parts

The three parts of a for loop header

```
for (int seat = 1; seat <= 4; seat++) {
```

Initialization

```
int seat = 1
```

Creates the loop control variable and gives it a starting value.

Boolean expression

```
seat <= 4
```

The test. The body runs only while it is true.

Update

```
seat++
```

Changes the variable after each pass, so the test can become false.

Same loop, one line instead of three

WHILE LOOP (TOPIC 2.7)

```
int seat = 1;
while (seat <= 4) {
    System.out.println("Seat " + seat);
    seat++;
}
```

FOR LOOP

```
for (int seat = 1; seat <= 4; seat++) {
    System.out.println("Seat " + seat);
}
```

OUTPUT

```
Seat 1
Seat 2
Seat 3
Seat 4
```



Both loops print the same output. A **for loop** is an iterative statement whose **header** holds the setup, the test, and the update in one place, so you can read how it counts at a glance.

Semicolons, not commas



The three parts of the header are separated by **two semicolons**. Commas there do not compile.

DOES NOT COMPILE

```
for (int seat = 1, seat <= 4, seat++) {  
    System.out.println("Seat " + seat);  
}
```

COMPILER SAYS

error: ';' expected

```
for (int seat = 1; seat <= 4; seat++) {  
    System.out.println("Seat " + seat);  
}
```

FIXED

A large, white, rounded square containing the number '2' in a bold, teal, sans-serif font.

2

PART 2

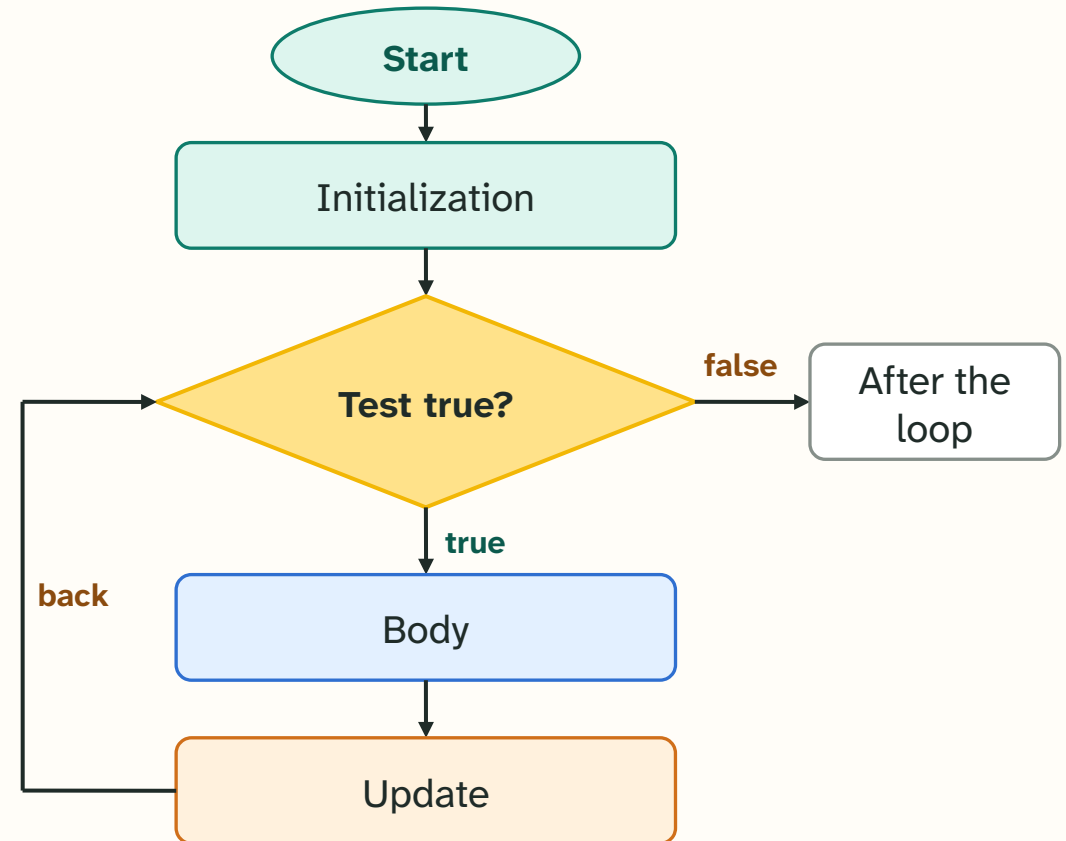
The order it runs in

Initialization once, then test, body, update

- 1 **Initialization**, once, before anything else.
- 2 **Test** the Boolean expression. False? The loop is over.
- 3 **Body**, the whole thing, top to bottom.
- 4 **Update**, only after the whole body.
- 5 **Back to the test** with the new value.



The test runs **one more time** than the body: the last test is the false one.



Every step of a three-day loop

```
for (int day = 1; day <= 3; day++) {  
    System.out.println("Day " + day);  
}  
System.out.println("Done");
```

OUTPUT

Day 1
Day 2
Day 3
Done

day	Test day <= 3	Body prints	Update
1	true	Day 1	day becomes 2
2	true	Day 2	day becomes 3
3	true	Day 3	day becomes 4
4	false: stop	(nothing)	(nothing)

Count: 1 initialization, 4 tests, 3 bodies, 3 updates.

Count the passes and the tests

PAUSE AND TRY

How many body runs? How many tests? Which p stops it?

```
for (int p = 5; p <= 20; p += 5) {  
    System.out.println(p);  
}
```

Question	Your answer
Times the body runs	
Times the test runs	
Value of p that stops it	

A semicolon after the header



A lone `;` is a statement that does nothing. After the header, it **becomes the body**, and the braces below are no longer part of the loop.

EXPECT THREE BEEPS?

```
for (int beep = 1; beep <= 3; beep++);  
{  
    System.out.println("Beep!");  
}
```

OUTPUT

Beep!

Only one. The loop ran 3 times with an empty body, then the block ran once.

Fix: delete the semicolon so the braces are the body again. It compiles either way, so check the output.



3

PART 3

Start, stop, step

Counting iterations

Header	Values of i	Iterations
<code>for (int i = 0; i < 4; i++)</code>	0, 1, 2, 3	4
<code>for (int i = 0; i <= 4; i++)</code>	0, 1, 2, 3, 4	5
<code>for (int i = 1; i <= 4; i++)</code>	1, 2, 3, 4	4

Initialization decides where counting starts.

Boolean expression decides where it stops.

Update decides the size and direction of each step.

Counting down, and stepping by more than 1

DOWN BY 4

```
for (int row = 9; row >= 1; row -= 4) {  
    System.out.println(row);  
}
```

OUTPUT

9
5
1

TIMES 3

```
for (int n = 3; n < 50; n *= 3) {  
    System.out.println(n);  
}
```

OUTPUT

3
9
27



Check the **first** and **last** value the body sees, and make sure the update moves toward making the test false.

The off by one error



Goal: print pages 1 through 5. `<` leaves out the stop value; `<=` includes it.

BUG

```
for (int page = 1; page < 5; page++) {  
    System.out.println("Page " + page);  
}
```

OUTPUT

```
Page 1  
Page 2  
Page 3  
Page 4
```

FIXED

```
for (int page = 1; page <= 5; page++) {  
    System.out.println("Page " + page);  
}
```

OUTPUT

```
Page 1  
Page 2  
Page 3  
Page 4  
Page 5
```

A large, bold, white number '4' is centered within a white rounded square, which is tilted slightly to the right.

PART 4

for and while: same loop, two outfits

Rewriting a for loop as a while loop

FOR LOOP

```
for (int temp = 50; temp > 20; temp -= 10) {  
    System.out.println(temp);  
}
```

WHILE LOOP

```
int temp = 50;  
while (temp > 20) {  
    System.out.println(temp);  
    temp -= 10;  
}
```

Initialization moves to a statement just before the loop.

Test becomes the while condition.

Update becomes the last statement of the body.

OUTPUT

Both print

50
40
30

The loop control variable lives in the loop



A variable declared in the header **stops existing** when the loop ends.

DOES NOT COMPILE

```
for (int lap = 1; lap <= 2; lap++) {  
    System.out.println("Lap " + lap);  
}  
System.out.println(lap);
```

COMPILER SAYS

```
error: cannot find symbol  
symbol:   variable lap
```

WORKS

```
int lap = 1;  
while (lap <= 2) {  
    System.out.println("Lap " + lap);  
    lap++;  
}  
System.out.println(lap);
```

OUTPUT

```
Lap 1  
Lap 2  
3
```

Which while loop matches?

PAUSE AND TRY

Which version prints exactly what the for loop prints?

```
for (int k = 10; k > 0; k -= 3) {  
    System.out.print(k + " ");  
}
```

VERSION A

```
int k = 10;  
while (k > 0) {  
    k -= 3;  
    System.out.print(k + " ");  
}
```

VERSION B

```
int k = 10;  
while (k > 0) {  
    System.out.print(k + " ");  
    k -= 3;  
}
```

Words to know

for loop

An iterative statement whose header holds the initialization, test, and update.

Initialization

First part of the header. Runs once, before the first test.

Boolean expression

Second part. The body runs only while it is true.

Update

Third part. Runs after each full pass of the body.

Loop control variable

Set up by the initialization, checked by the test, changed by the update.

Iteration

One run of the loop body.

Say the order out loud



Explain to your partner the order in which a for loop runs its three parts and its body.

SENTENCE STARTER

First the _____ runs once. Then the loop repeats _____, _____, _____ until the test is false.

Next: the Start, Stop, Step puzzle worksheet.