

Building Java Programs

Chapter 2

Lecture 2-2: The `for` Loop

reading: 2.3

self-check: 12-26

exercises: 2-14

videos: Ch. 2 #3

Modify-and-assign operators

shortcuts to modify a variable's value

Shorthand

variable += **value**;

variable -= **value**;

variable *= **value**;

variable /= **value**;

variable %= **value**;

Equivalent longer version

variable = **variable** + **value**;

variable = **variable** - **value**;

variable = **variable** * **value**;

variable = **variable** / **value**;

variable = **variable** % **value**;

```
x += 3;
```

```
gpa -= 0.5;
```

```
number *= 2;
```

```
// x = x + 3;
```

```
// gpa = gpa - 0.5;
```

```
// number = number * 2;
```

Increment and decrement

shortcuts to increase or decrease a variable's value by 1

Shorthand

variable++;

variable--;

```
int x = 2;
```

```
x++;
```

```
double gpa = 2.5;
```

```
gpa--;
```

Equivalent longer version

variable = **variable** + 1;

variable = **variable** - 1;

```
// x = x + 1; (or x += 1; )
```

```
// x now stores 3
```

```
// gpa -= 1;
```

```
// gpa now stores 1.5
```

Repetition over a range

```
System.out.println(1 + " squared = " + 1 * 1);  
System.out.println(2 + " squared = " + 2 * 2);  
System.out.println(3 + " squared = " + 3 * 3);  
System.out.println(4 + " squared = " + 4 * 4);  
System.out.println(5 + " squared = " + 5 * 5);  
System.out.println(6 + " squared = " + 6 * 6);
```

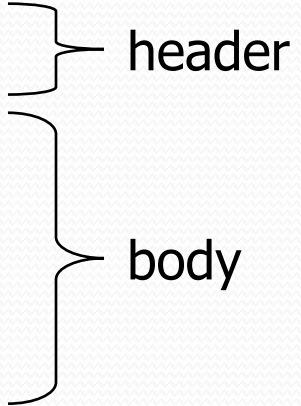
- Intuition: "I want to print a line for each number from 1 to 6"
- There's a statement, the `for` loop, that does just that!

```
for (int i = 1; i <= 6; i++) {  
    System.out.println(i + " squared = " + (i * i));  
}
```

- "For each integer `i` from 1 through 6, print ..."

for loop syntax

```
for (initialization; test; update) {  
    statement;  
    statement;  
    ...  
    statement;  
}
```



- Perform **initialization** once.
- Repeat the following:
 - Check if the **test** is true. If not, stop.
 - Execute the **statements**.
 - Perform the **update**.

Initialization

```
for (int i = 1; i <= 6; i++) {  
    System.out.println(i + " squared = " + (i * i));  
}
```

- Tells Java what variable to use in the loop
 - Called a *loop counter*
 - Can use any variable name, not just *i*
 - Can start at any value, not just 1

Test

```
for (int i = 1; i <= 6; i++) {  
    System.out.println(i + " squared = " + (i * i));  
}
```

- Tests whether the loop should stop
 - Typically uses *comparison operators*:
 - < less than
 - <= less than or equal to
 - > greater than
 - >= greater than or equal to

Update

```
for (int i = 1; i <= 6; i++) {  
    System.out.println(i + " squared = " + (i * i));  
}
```

- What to do after the loop body
 - Update the loop-counter variable appropriately
 - Without an update, you would have an *infinite loop*
- Can be any expression:

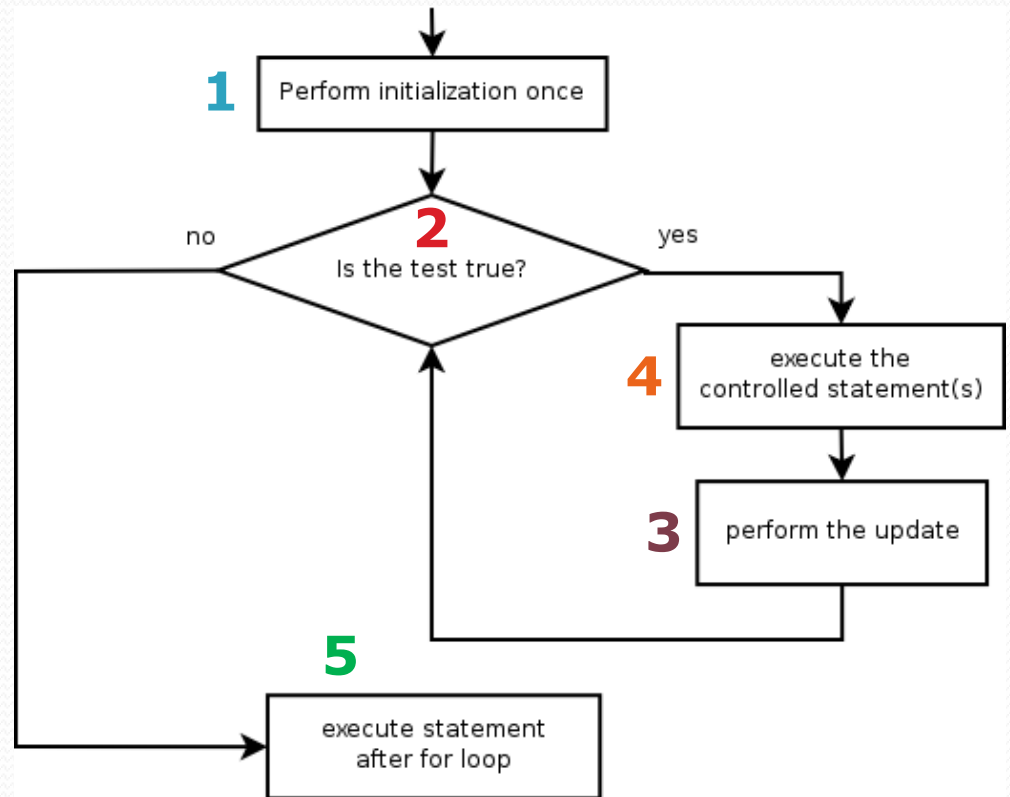
```
for (int i = 1; i <= 9; i += 2) {  
    System.out.println(i);  
}
```


Loop walkthrough

```
1 for (int i = 1; i <= 4; i++) {  
  2  
  3  
  4 System.out.println(i + " squared = " + (i * i));  
}  
5 System.out.println("Whoo!");
```

Output:

```
1 squared = 1  
2 squared = 4  
3 squared = 9  
4 squared = 16  
Whoo!
```



General repetition

```
System.out.println("I am so smart");  
System.out.println("I am so smart");  
System.out.println("I am so smart");  
System.out.println("I am so smart");  
System.out.println("I am so smart");  
System.out.println("S-M-R-T");  
System.out.println("I mean S-M-A-R-T");
```

- The loop's body doesn't have to use the counter variable:

```
for (int i = 1; i <= 5; i++) { // repeat 5 times  
    System.out.println("I am so smart");  
}  
System.out.println("S-M-R-T");  
System.out.println("I mean S-M-A-R-T");
```

Multi-line loop body

Output:

```
+-----+
\       /
/       \
\       /
/       \
\       /
/       \
+-----+
```

```
System.out.println("+-----+");
for (int i = 1; i <= 3; i++) {
    System.out.println("\\       /");
    System.out.println("/       \\");
}
System.out.println("+-----+");
```

Expressions for counter

```
int highTemp = 5;  
for (int i = -3; i <= highTemp / 2; i++) {  
    System.out.println(i * 1.8 + 32);  
}
```

- Output:

26.6
28.4
30.2
32.0
33.8
35.6

System.out.print

- Prints without moving to a new line
 - allows you to print partial messages on the same line

```
int highestTemp = 5;
for (int i = -3; i <= highestTemp / 2; i++) {
    System.out.print((i * 1.8 + 32) + " ");
}
```

- Output:

26.6 28.4 30.2 32.0 33.8 35.6

Counting down

- The **update** can use `--` to make the loop count down.
 - Be sure to use the right **test** (`>` or `>=` instead of `<` or `<=`)

```
System.out.print("T-minus ");
for (int i = 10; i >= 1; i--) {
    System.out.print(i + ", ");
}
System.out.println("blastoff!");
```

- **Output:**

```
T-minus 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, blastoff!
```

Where are we

- Done: many basic features of Java
 - Static methods
 - `int`, `double`, and strings
 - Expressions: `+`, `-`, `*`, `/`, `%`, `<`, `<=`, `>`, `>=`
 - Variables
 - For loops
 - `System.out.println` and `System.out.print`
- Many more features to come, but first how to use for loops effectively
 - No new rules, just new *programming patterns*
 - And practice designing programs
 - For loops can *nest* (be inside other for loops)

Mapping loops to numbers

```
for (int count = 1; count <= 5; count++) {  
    ...  
}
```

- What statement in the body would cause the loop to print:
4 7 10 13 16

```
for (int count = 1; count <= 5; count++) {  
    System.out.print(3 * count + 1 + " ");  
}
```


Loop tables

- What statement in the body would cause the loop to print:
2 7 12 17 22
- To see patterns, make a table of `count` and the numbers.
 - Each time `count` goes up by 1, the number should go up by 5.
 - But `count * 5` is too great by 3, so we subtract 3.

count	number to print	<code>5 * count</code>	<code>5 * count - 3</code>
1	2	5	2
2	7	10	7
3	12	15	12
4	17	20	17
5	22	25	22

Loop tables question

- What statement in the body would cause the loop to print:
17 13 9 5 1
- You try it...
 - Each time `count` goes up 1, the number printed should ...
 - But this multiple is off by a margin of ...

count	number to print	$-4 * \text{count}$	$-4 * \text{count} + 21$
1	17	-4	17
2	13	-8	13
3	9	-12	9
4	5	-16	5
5	1	-20	1

Nested loops

reading: 2.3

self-check: 22-26

exercises: 10-14

videos: Ch. 2 #4

Redundancy between loops

```
for (int j = 1; j <= 5; j++) {  
    System.out.print(j + "\t");  
}  
System.out.println();  
  
for (int j = 1; j <= 5; j++) {  
    System.out.print(2 * j + "\t");  
}  
System.out.println();  
  
for (int j = 1; j <= 5; j++) {  
    System.out.print(3 * j + "\t");  
}  
System.out.println();  
  
for (int j = 1; j <= 5; j++) {  
    System.out.print(4 * j + "\t");  
}  
System.out.println();
```

Output:

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20

Nested loops

- **nested loop:** A loop placed inside another loop.

```
for (int i = 1; i <= 4; i++) {  
    for (int j = 1; j <= 5; j++) {  
        System.out.print((i * j) + "\t");  
    }  
    System.out.println(); // to end the line  
}
```

- **Output:**

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20

- Statements in the outer loop's body are executed 4 times.
 - The inner loop prints 5 numbers each time it is run.

Nested for loop exercise

- What is the output of the following nested for loops?

```
for (int i = 1; i <= 6; i++) {  
    for (int j = 1; j <= 10; j++) {  
        System.out.print("*");  
    }  
    System.out.println();  
}
```

- Output:

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

Nested for loop exercise

- What is the output of the following nested for loops?

```
for (int i = 1; i <= 6; i++) {  
    for (int j = 1; j <= i; j++) {  
        System.out.print("*");  
    }  
    System.out.println();  
}
```

- Output:

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

Nested for loop exercise

- What is the output of the following nested for loops?

```
for (int i = 1; i <= 6; i++) {  
    for (int j = 1; j <= i; j++) {  
        System.out.print(i);  
    }  
    System.out.println();  
}
```

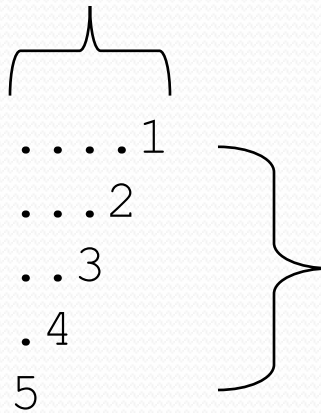
- Output:

```
1  
22  
333  
4444  
55555  
666666
```


Complex lines

- What nested `for` loops produce the following output?

inner loop (repeated characters on each line)



```
.....1
...2
..3
.4
5
```

outer loop (loops 5 times because there are 5 lines)

- Can build multiple complex lines of output using:
 - an *outer "vertical" loop* for each of the lines
 - *inner "horizontal" loop(s)* for the patterns within each line

Outer and inner loop

- First write the outer loop, from 1 to the number of lines.

```
for (int line = 1; line <= 5; line++) {  
    ...  
}
```

- Now look at the line contents. Each line has a pattern:
 - some dots (0 dots on the last line)
 - a number

```
....1  
...2  
..3  
.4  
5
```

Nested for loop exercise

- Make a table to represent any patterns on each line.

```
.....1
....2
...3
..4
.5
```

line	# of dots	$-1 * \text{line}$	$-1 * \text{line} + 5$
1	4	-1	4
2	3	-2	3
3	2	-3	2
4	1	-4	1
5	0	-5	0

- To print a character multiple times, use a for loop.

```
for (int j = 1; j <= 4; j++) {
    System.out.print(".");           // 4 dots
}
```

Nested for loop solution

- Answer:

```
for (int line = 1; line <= 5; line++) {  
    for (int j = 1; j <= (-1 * line + 5); j++) {  
        System.out.print(".");  
    }  
    System.out.println(line);  
}
```

- Output:

```
.....1  
...2  
..3  
.4  
5
```

Nested for loop exercise

- What is the output of the following nested for loops?

```
for (int line = 1; line <= 5; line++) {  
    for (int j = 1; j <= (-1 * line + 5); j++) {  
        System.out.print(".");  
    }  
    for (int k = 1; k <= line; k++) {  
        System.out.print(line);  
    }  
    System.out.println();  
}
```

- Answer:

```
.....1  
...22  
..333  
.4444  
55555
```

Nested for loop exercise

- Modify the previous code to produce this output:

```
.....1
...2.
..3..
.4...
5.....
```

- Answer:

```
for (int line = 1; line <= 5; line++) {
    for (int j = 1; j <= (-1 * line + 5); j++) {
        System.out.print(".");
    }
    System.out.print(line);
    for (int j = 1; j <= (line - 1); j++) {
        System.out.print(".");
    }
    System.out.println();
}
```

Common errors

- Both of the following sets of code produce *infinite loops*:

```
for (int i = 1; i <= 10; i++) {  
    for (int j = 1; i <= 5; j++) {  
        System.out.print(j);  
    }  
    System.out.println();  
}
```

```
for (int i = 1; i <= 10; i++) {  
    for (int j = 1; j <= 5; i++) {  
        System.out.print(j);  
    }  
    System.out.println();  
}
```