

A brick wall on the left side of a blue background. The bricks are reddish-brown with white mortar. The wall is partially visible, extending from the left edge towards the center of the frame.

# Building Java Programs

## Chapter 2: Primitive Data and Definite Loops

# Lecture outline

- repetition
  - the `for` loop
  - nested loops

A brick wall on the left side of a blue background. The bricks are reddish-brown with white mortar. The wall is partially visible, extending from the left edge towards the center. The background is a solid, light blue color.

# The `for` loop

reading: 2.3

# Repetition with `for` loops

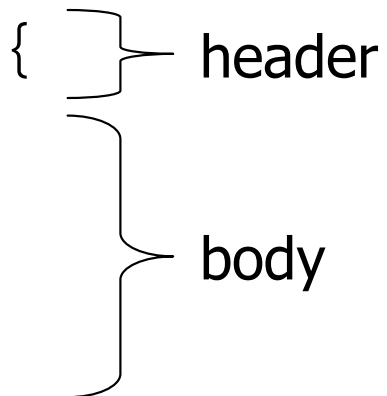
- So far, when we wanted to perform a task multiple times, we have written redundant code:
  - ```
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("I am so smart");
System.out.println("S-M-R-T");
System.out.println("I mean S-M-A-R-T");
```
- Java has a statement called a *for loop statement* that instructs the computer to perform a task many times.
  - ```
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");
```

# for loop syntax

- **for loop**: A Java statement that executes a group of statements repeatedly until a given test fails.

- General syntax:

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



- Example:

```
for (int i = 1; i <= 10; i++) {  
    System.out.println("His name is Robert Paulson");  
}
```

# for loop over range of ints

- We'll write `for` loops over integers in a given range.
  - The loop declares a *loop counter* variable that is used in the test, update, and body of the loop.

```
for (int <name> = 1; <name> <= <value>; <name>++)
```

- Example:

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

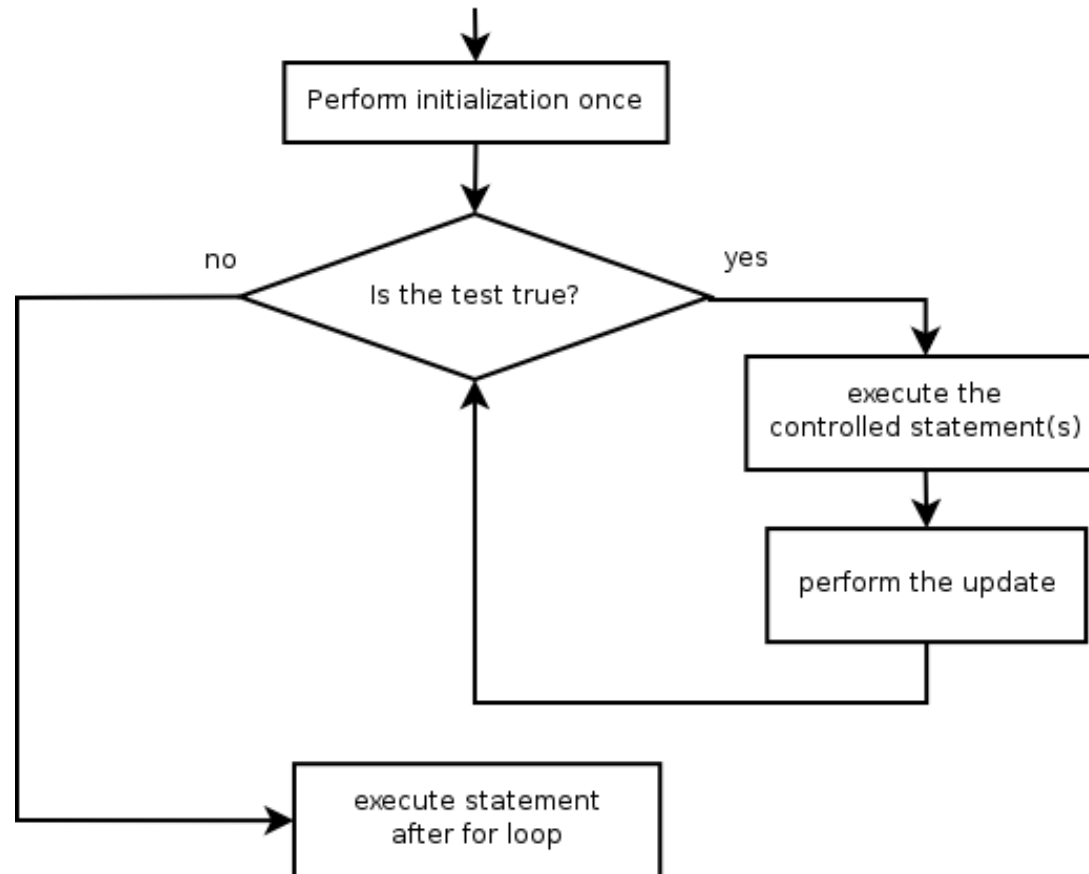
- Interpretation: "For each integer **i** from 1 through 6, ..."

- Output:

```
1 squared is 1  
2 squared is 4  
3 squared is 9  
4 squared is 16  
5 squared is 25  
6 squared is 36
```

# for loop flow diagram

- Behavior of the `for` loop:
  - Start out by performing the **<initialization>** once.
  - Repeatedly execute the **<statement(s)>** followed by the **<update>** as long as the **<test>** is still a true statement.



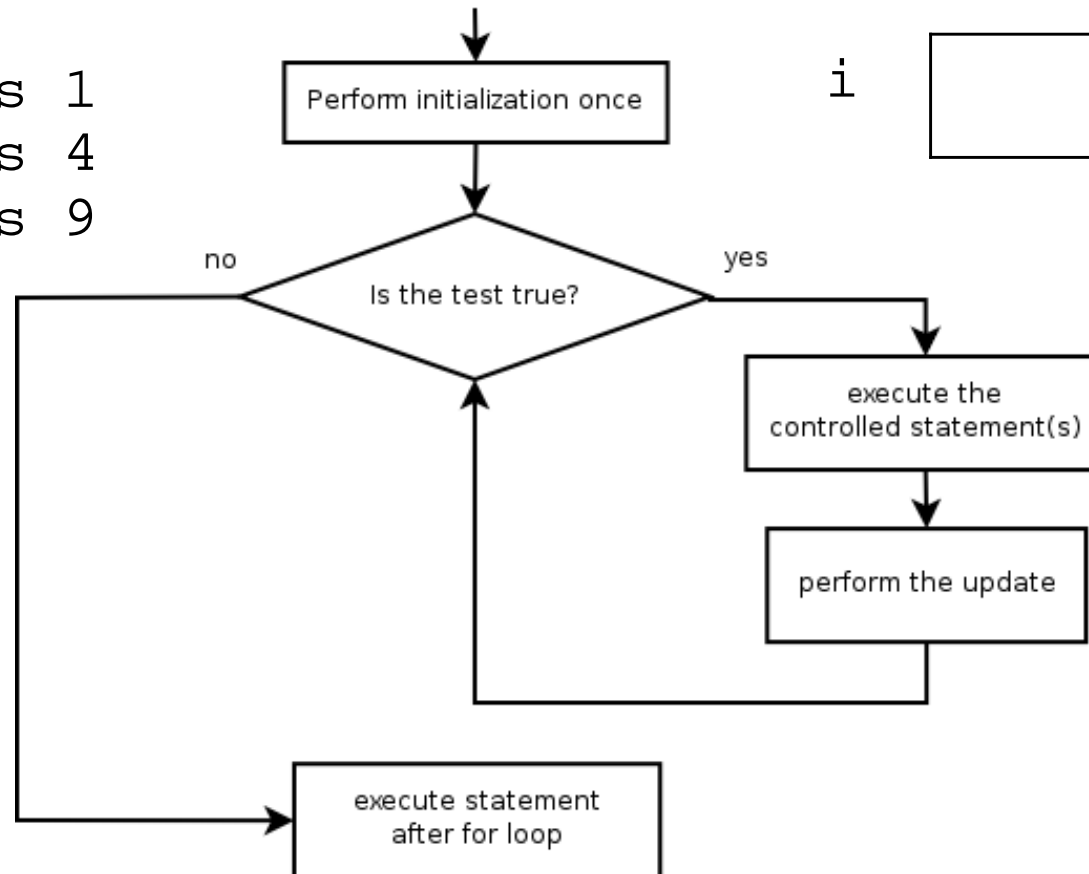
# Loop walkthrough

Let's walk through the following for loop:

```
for (int i = 1; i <= 3; i++) {  
    System.out.println(i + " squared is " + (i * i));  
}
```

Output:

```
1 squared is 1  
2 squared is 4  
3 squared is 9
```





# Another example for loop

- The body of a `for` loop can contain multiple lines.
  - Example:

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

- Output:

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

# Some for loop variations

- The initial and final values for the loop counter variable can be arbitrary numbers or expressions:

- Example:

```
for (int i = -3; i <= 2; i++) {  
    System.out.println(i);  
}
```

- Output:

```
-3  
-2  
-1  
0  
1  
2
```

- Example:

```
for (int i = 1 + 3 * 4; i <= 5248 % 100; i++) {  
    System.out.println(i + " squared is " + (i * i));  
}
```

# Downward-counting for loop

- The update can also be a `--` or other operator, to make the loop count down instead of up.
  - This also requires changing the test to say `>=` instead of `<=` .

```
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!
```

# Single-line for loop

- When a `for` loop only has one statement in its body, the `{ }` braces may be omitted.

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

- However, this can lead to mistakes where a line appears to be inside a loop, but is not:

```
■ for (int i = 1; i <= 3; i++)  
    System.out.println("This is printed 3 times");  
    System.out.println("So is this... or is it?");
```

- Output:

```
This is printed 3 times  
This is printed 3 times  
This is printed 3 times  
So is this... or is it?
```

# for loop questions

- Write a loop that produces the following output.

```
On day #1 of Christmas, my true love sent to me
```

```
On day #2 of Christmas, my true love sent to me
```

```
On day #3 of Christmas, my true love sent to me
```

```
On day #4 of Christmas, my true love sent to me
```

```
On day #5 of Christmas, my true love sent to me
```

```
...
```

```
On day #12 of Christmas, my true love sent to me
```

- Write a loop that produces the following output.

```
2 4 6 8
```

```
Who do we appreciate
```

# Mapping loops to numbers

- Suppose that we have the following loop:

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

- What statement could we write in the body of the loop that would make the loop print the following output?

3 6 9 12 15

- Answer:

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

# Mapping loops to numbers 2

- Now consider another loop of the same style:

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

- What statement could we write in the body of the loop that would make the loop print the following output?

```
4 7 10 13 16
```

- Answer:

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

# Loop number tables

- What statement could we write in the body of the loop that would make the loop print the following output?

2 7 12 17 22

- To find the pattern, it can help to make a table of the count and the number to print.
  - Each time count goes up by 1, the number should go up by 5.
  - But  $\text{count} * 5$  is too great by 3, so we must subtract 3.

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



# Loop table question

- What statement could we write in the body of the loop that would make the loop print the following output?

17 13 9 5 1

- Let's create the loop table together.
  - Each time count goes up 1, the number should ...
  - But this multiple is off by a margin of ...

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

# Degenerate loops

- Some loops execute 0 times, because of the nature of their test and update.

```
// a degenerate loop
for (int i = 10; i < 5; i++) {
    System.out.println("How many times do I print?");
}
```

- Some loops execute endlessly (or far too many times), because the loop test never fails.
- A loop that never terminates is called an *infinite loop*.

```
for (int i = 10; i >= 1; i++) {
    System.out.println("Runaway Java program!!!");
}
```

# Nested loops

- **nested loop:** Loops placed inside one another.
  - The inner loop's counter variable must have a different name.

```
for (int i = 1; i <= 3; i++) {  
    System.out.println("i = " + i);  
    for (int j = 1; j <= 2; j++) {  
        System.out.println("    j = " + j);  
    }  
}
```

Output:

```
i = 1  
    j = 1  
    j = 2  
i = 2  
    j = 1  
    j = 2  
i = 3  
    j = 1  
    j = 2
```

# More nested loops

- In this example, all of the statements in the outer loop's body are executed 5 times.
  - The inner loop prints 10 numbers each of those 5 times, for a total of 50 numbers printed.

```
for (int i = 1; i <= 5; i++) {  
    for (int j = 1; j <= 10; j++) {  
        System.out.print((i * j) + " ");  
    }  
    System.out.println(); // to end the line  
}
```

## Output:

```
1 2 3 4 5 6 7 8 9 10  
2 4 6 8 10 12 14 16 18 20  
3 6 9 12 15 18 21 24 27 30  
4 8 12 16 20 24 28 32 36 40  
5 10 15 20 25 30 35 40 45 50
```

# 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
```

# Nested for loop exercise

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

1, 1

2, 1

3, 1

1, 2

2, 2

3, 2

- Answer:

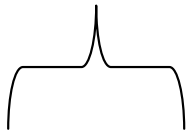
```
for (int y = 1; y <= 2; y++) {  
    for (int x = 1; x <= 3; x++) {  
        System.out.println(x + ", " + y);  
    }  
}
```



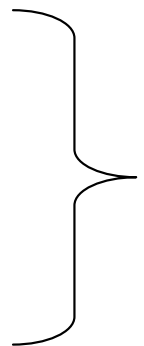
# Nested for loop exercise

- 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)

- This is an example of a nested loop problem where we build multiple complex lines of output:
  - outer "vertical" loop for each of the lines
  - inner "horizontal" loop(s) for the patterns within each line

# Nested for loop exercise

- First we write the outer loop, which always goes from 1 to the number of lines desired:

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

- We notice that each line has the following pattern:
  - some number of dots (0 dots on the last line)
  - a number

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

# Nested for loop exercise

- Next we make a table to represent any necessary patterns on that line:

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

line	# of dots	value displayed	
1	4	1	
2	3	2	
3	2	3	
4	1	4	
5	0	5	

- Answer:

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

# Nested for loop exercise

- A `for` loop can have more than one loop nested in it.
- What is the output of the following nested `for` loops?

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

line	# of dots	value displayed	# of dots
1	4	1	0
2	3	2	1
3	2	3	2
4	1	4	3
5	0	5	4

- 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 nested loop bugs

- It is easy to accidentally type the wrong loop variable.

- What is the output of the following nested loops?

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

- What is the output of the following nested loops?

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

# How to comment: for loops

- Place a comment on complex loops explaining *what* they do conceptually, not the mechanics of the syntax.

- Bad:

```
// This loop repeats 10 times, with i from 1 to 10.
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 5; j++) { // loop goes 5 times
        System.out.print(j); // print the j
    }
    System.out.println();
}
```

- Better:

```
// Prints 12345 ten times on ten separate lines.
for (int i = 1; i <= 10; i++) {
    for (int j = 1; j <= 5; j++) {
        System.out.print(j);
    }
    System.out.println(); // end the line of output
}
```