Building Java Programs

Chapter 2: Primitive Data and Definite Loops

Lecture outline

- repetition
  - the `for` loop
  - nested loops

The big picture

- Computers excel at redundant tasks
  - Loops give us syntax for writing them concisely
  - Without loops, computers can't do more than programmers

- Endless applications
  - A robot repeats behavior until it has found something
  - Check my e-mail until one comes in
  - Format all the messages on the message board

The for loop

reading: 2.3

self-check: 12-21

exercises: 2-9

Repetition with `for` loops

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

- Intuition: "Java, print a line for each integer 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 is " + (i * i));
}
```

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

for loop syntax

- **`for` loop**: A Java statement that executes a group of statements repeatedly as long as a given test is true
  - General syntax:

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

header

body

  - 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.

Initialization

- Tells Java what variable to use within the loop
  - Called *loop counter*

- Can either declare new variable or use existing one
  - Variables declared in loop initialization disappear after loop

```
for (int i = 1; i <= 6; i++) {
      System.out.println(i + " squared is " + (i * i));
}
// ERROR: variable i doesn't exist
System.out.println("Biggest number printed: " + i);
```

Test

- Generally tests loop counter against a bound

- Uses familiar comparison operators

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

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

Update

- The loop counter has to change at each repetition
  - Otherwise: infinite loop!
- Can be any expression

```
for (int i = 2; i <= 8; i = i + 2) {
      System.out.println(i);
}
```

  - Output:

```
2
4
6
8
```

Increment and decrement

- Adding or subtracting 1 is common
- Shortcut: the *increment* and *decrement* operators

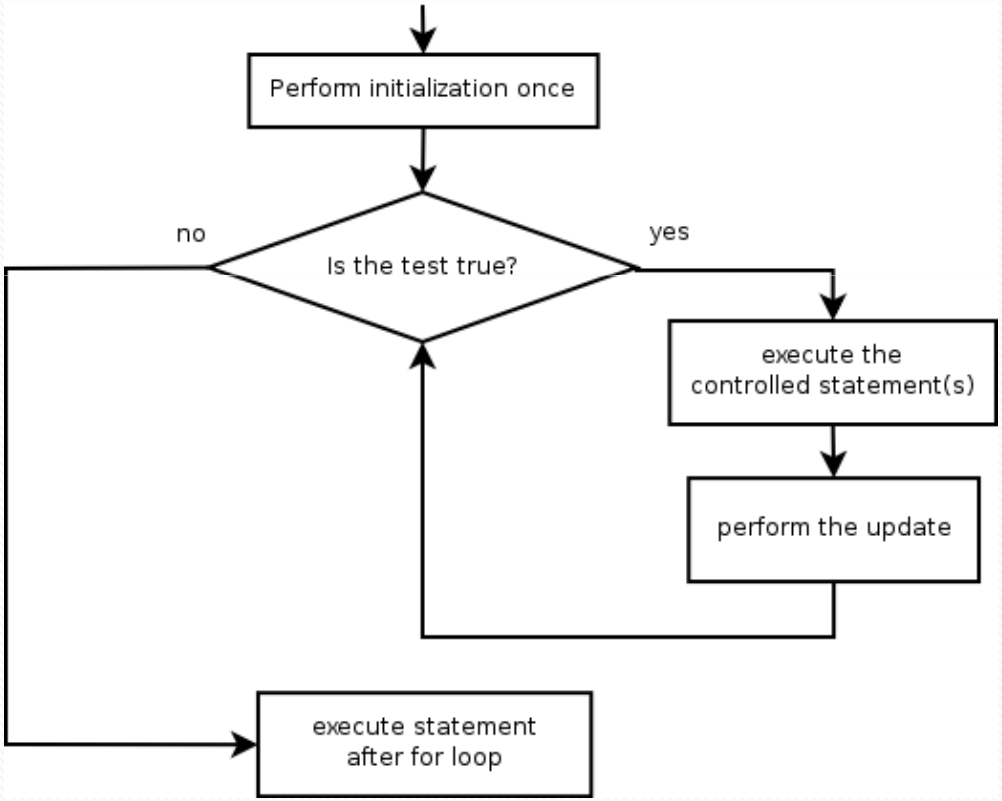
Shorthand	Equivalent longer version
<variable> `++ ;`	***<variable>*** `=` ***<variable>*** `+ 1;`
<variable> `-- ;`	***<variable>*** `=` ***<variable>*** `- 1;`

- Examples:

```
int x = 2;
x++;                          // x = x + 1;
                              // x now stores 3

double gpa = 2.5;
gpa--;                        // gpa = gpa - 1;
                              // gpa now stores 1.5
```

for loop flow diagram



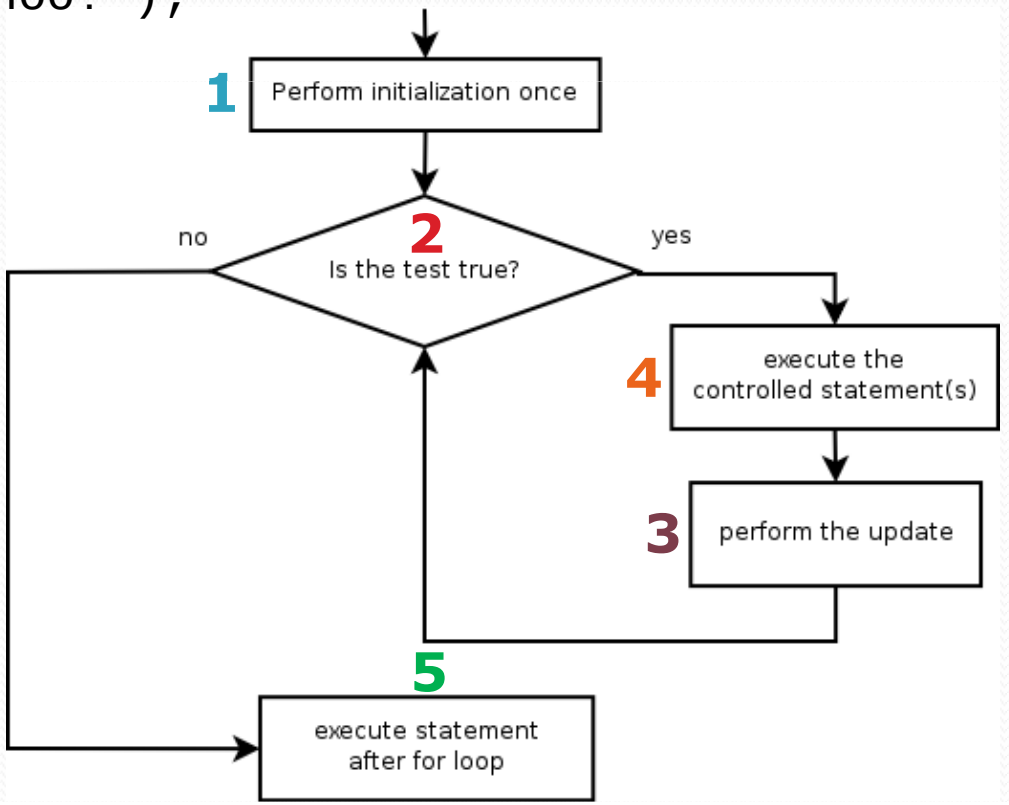
Loop walkthrough

Let's walk through the following `for` loop:

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

Output:

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



Repetition with for loops

```
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 doesn't have to use the counter explicitly
  - ```
    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");
    ```
    ```

Multiple line `for` loops

- 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:

```
int highestTemp = 7;
for (int i = -3; i <= highestTemp / 2; i++) {
    System.out.println(i * 1.8 + 32);
}
```

  - Output:

```
26.6
28.4
30.2
32.0
33.8
35.6
37.4
```

System.out.print

- Recall: `System.out.println` prints a line of output and then advances to a new line.
- `System.out.print` prints without moving to a new line.
  - This allows you to print partial messages on the same line.
- Example:

```
int highestTemp = 7;
for (int i = -3; i <= highestTemp / 2; i++) {
    System.out.print("\t" + (i * 1.8 + 32));
}
```

Output:

```
    26.6    28.4    30.2    32.0    33.8    35.6    37.4
```

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!
```

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?

```
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 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

Nested loops

reading: 2.3

self-check: 22-26

exercises: 10-14

Redundancy in loops

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

- Rewrite the multiplications example to reduce redundancy

```
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 of those 4 times, for a total of 20 numbers printed.

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();
}
```

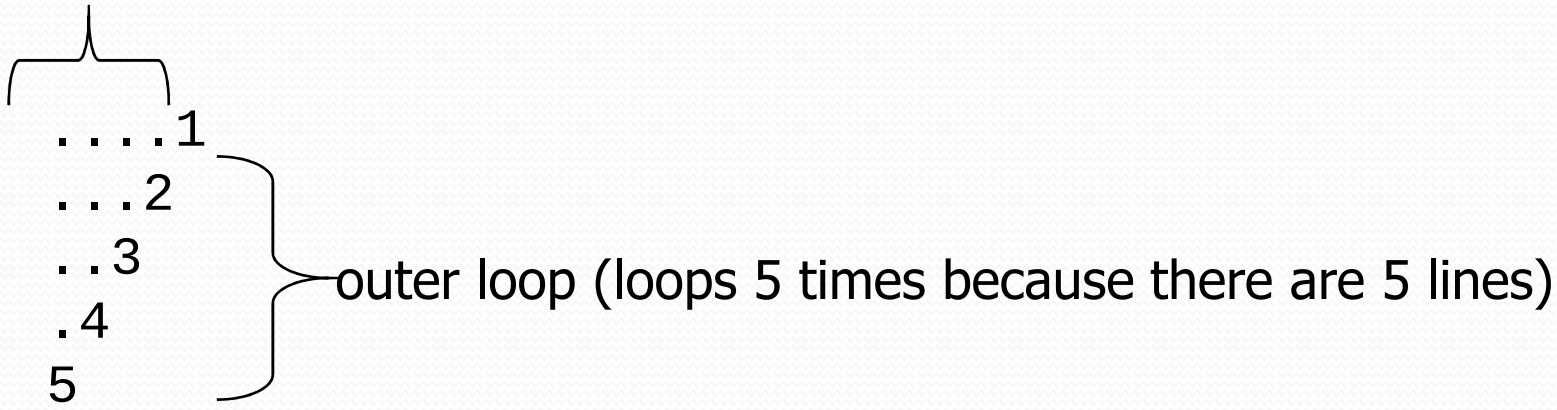
- Output:

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

Nested `for` loop exercise

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

inner loop (repeated characters on each line)



- 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
5
```

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

- 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
}
```