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
Increment and decrement

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

**Shorthand**

```plaintext
variable++;
variable--;
```

**Equivalent longer version**

```plaintext
variable = variable + 1;
variable = variable - 1;
```

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

double gpa = 2.5;
gpa--; // gpa = gpa - 1;
// gpa now stores 1.5
```
# Modify-and-assign operators

*shortcuts to modify a variable's value*

<table>
<thead>
<tr>
<th>Shorthand</th>
<th>Equivalent longer version</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>variable += value;</code></td>
<td><code>variable = variable + value;</code></td>
</tr>
<tr>
<td><code>variable -= value;</code></td>
<td><code>variable = variable - value;</code></td>
</tr>
<tr>
<td><code>variable *= value;</code></td>
<td><code>variable = variable * value;</code></td>
</tr>
<tr>
<td><code>variable /= value;</code></td>
<td><code>variable = variable / value;</code></td>
</tr>
<tr>
<td><code>variable %= value;</code></td>
<td><code>variable = variable % value;</code></td>
</tr>
</tbody>
</table>

- `x += 3;`  
  `// x = x + 3;`
- `gpa -= 0.5;`  
  `// gpa = gpa - 0.5;`
- `number *= 2;`  
  `// number = number * 2;`
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
Tests the loop counter variable against a bound

- Tests the loop counter variable against a bound
- 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));
}

- Changes loop counter's value after each repetition
  - 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

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

Output:

1 squared = 1
2 squared = 4
3 squared = 9
4 squared = 16
Whoo!
General repetition

```java
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:

```java
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

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

• Output:
  +-----+
     /\ 
     / / 
     / / 
     / / 
     / / 
/+-----+
Expressions for counter

```java
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

```java
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.
  - The **test** must say `>` instead of `<

```java
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

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 + " ");
}
Slope-intercept

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

- What statement in the body would cause the loop to print:
  2 7 12 17 22

- Much like a slope-intercept problem:
  - count is x
  - the printed number is y
  - The line passes through points:
    (1, 2), (2, 7), (3, 12), (4, 17), (5, 22)
  - What is the equation of the line?
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.

<table>
<thead>
<tr>
<th>count</th>
<th>number to print</th>
<th>5 * count</th>
<th>5 * count - 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>20</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>22</td>
<td>25</td>
<td>22</td>
</tr>
</tbody>
</table>
Loop tables question

- What statement in the body would cause the loop to print: 17 13 9 5 1

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

<table>
<thead>
<tr>
<th>count</th>
<th>number to print</th>
<th>-4 * count</th>
<th>-4 * count + 21</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>17</td>
<td>-4</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>-8</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>-12</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>-16</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>-20</td>
<td>1</td>
</tr>
</tbody>
</table>
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();
Nested loops

- **nested loop**: A loop placed inside another loop.
  
  ```java
  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?

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

- Output:

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

Copyright 2008 by Pearson Education
Nested for loop exercise

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

```java
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?

```java
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?

  \[ \text{inner loop (repeated characters on each line)} \]

  \[
  \begin{align*}
  \ldots 1 \\
  \ldots 2 \\
  \ldots 3 \\
  \ldots 4 \\
  5
  \end{align*}
  \]

  \[ \text{outer loop (loops 5 times because there are 5 lines)} \]

- We must 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.
  ```java
  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.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td># of dots</td>
<td>-1 * line</td>
<td>-1 * line + 5</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>-1</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>-2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>-3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>-4</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>-5</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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

- **Answer:**
  
  ```java
  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?
  ```java
  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:
  ```diff
  ....1
  ...22
  ..333
  .4444
  55555
  ```
Nested for loop exercise

- Modify the previous code to produce this output:
  
  
  . ....1
  ...2.
  ..3..
  .4...
  5....

- Answer:

```java
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*:

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

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