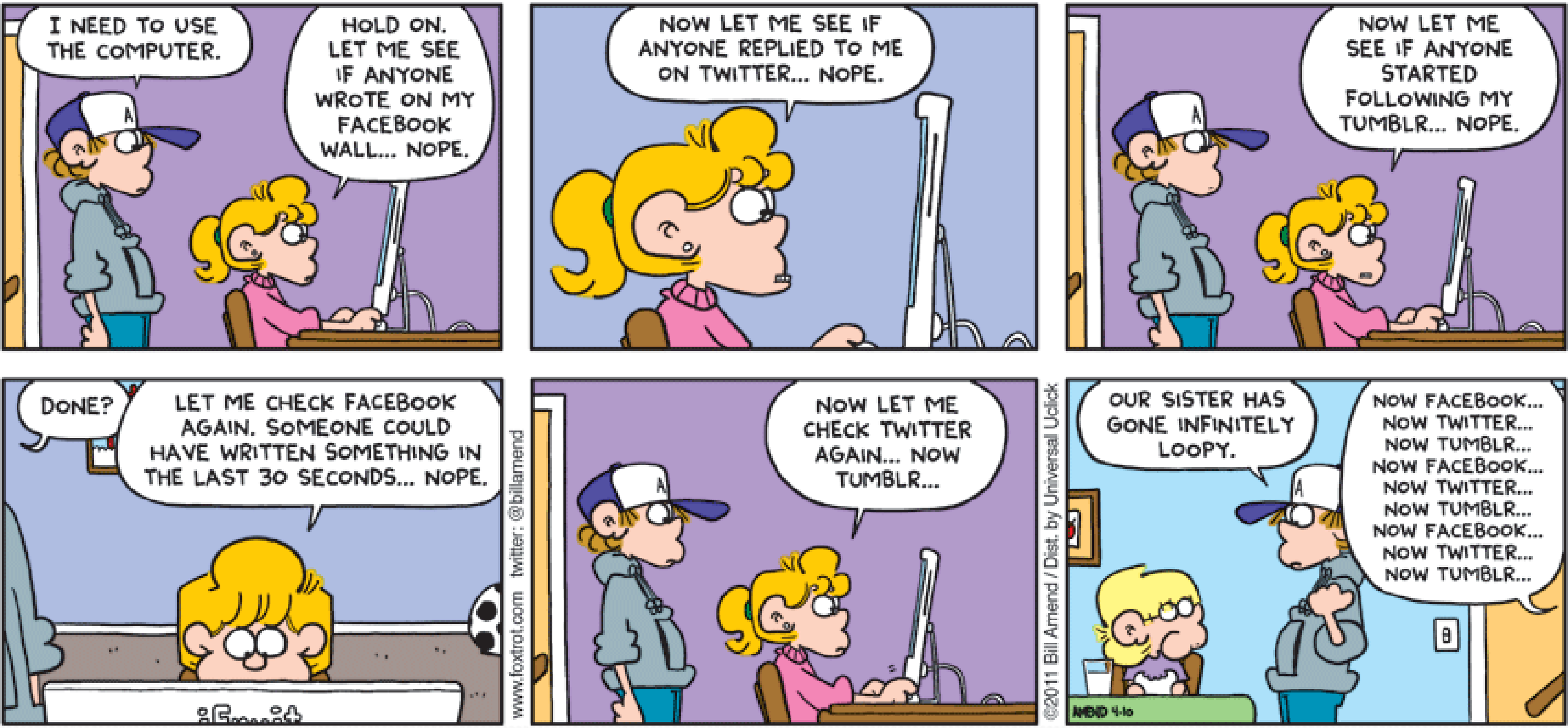
Building Java Programs

Chapter 2

Lecture 2-3: The `for` Loop

reading: 2.3


I NEED TO USE THE COMPUTER.
HOLD ON. LET ME SEE IF ANYONE WROTE ON MY FACEBOOK WALL... NOPE.
NOW LET ME SEE IF ANYONE REPLIED TO ME ON TWITTER... NOPE.
NOW LET ME SEE IF ANYONE STARTED FOLLOWING MY TUMBLR... NOPE.
DONE?
LET ME CHECK FACEBOOK AGAIN. SOMEONE COULD HAVE WRITTEN SOMETHING IN THE LAST 30 SECONDS... NOPE.
NOW LET ME CHECK TWITTER AGAIN... NOW TUMBLR...
OUR SISTER HAS GONE INFINITELY LOOPY.
NOW FACEBOOK... NOW TWITTER... NOW TUMBLR... NOW FACEBOOK... NOW TWITTER... NOW TUMBLR... NOW FACEBOOK... NOW TWITTER... NOW TUMBLR...
AMEND 4-10

www.foxtrot.com twitter: @billamend

©2011 Bill Amend / Dist. by Universal Uclick

Repetition with `for` loops

- We have seen repeating a statement is redundant/not good:

```
System.out.println("Homer says:");
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... I mean S-M-A-R-T");
```

- Java's **`for` loop** statement performs a task many times.

```
System.out.println("Homer says:");
for (int i = 1; i <= 4; i++) {    // repeat 4 times
    System.out.println("I am so smart");
}
System.out.println("S-M-R-T... I mean S-M-A-R-T");
```

for loop

- Known as a “definite” (as opposed to “indefinite”) loop
- Typically used when you know how many times you need to loop
    - …do “that” 5 times
    - …print the first 10 prime numbers
    - etc.

Control structures

- **Control structure**: a programming construct that affects the flow of a program's execution, and controls other statements

- Controlled code may include one or more statements

- The for loop is an example of a looping control structure

for loop syntax

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

header

body

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

Initialization

```
for (int i = 1; i <= 4; i++) {
    System.out.println("I am so smart");
}
```

Initialization

```
for (int i = 1; i <= 4; i++) {
    System.out.println("I am so smart");
}
```

- Tells Java what variable to use in the loop
  - The variable is called a *loop counter*
    - can use any name, not just `i (but i is the most common)`
    - can start at any value, not just `1 (start at zero…)`
    - only valid in the loop
  - Performed once as the loop begins

Test

```
for (int i = 1; i <= 4; i++) {
    System.out.println("I am so smart");
}
```

Test

```
for (int i = 1; i <= 4; i++) {
    System.out.println("I am so smart");
}
```

- Tests the loop counter variable against a limit
  - Uses comparison operators:

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

- Performed at the start of every loop iteration

Update

```
for (int i = 1; i <= 4; i++) {
    System.out.println("I am so smart");
}
```

Update

```
for (int i = 1; i <= 4; i++) {
    System.out.println("I am so smart");
}
```

- Updates the loop counter accounting for each iteration
  - Is an expression that changes the loop counter
    - Increment by one
    - Decrement by one
    - Increment or decrement by two
    - Etc …
- Performed at the end (last step) of each loop iteration

Increment and decrement

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

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

```
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

Shorthand	Equivalent longer version
variable += **exp**;	**variable** = **variable** + (**exp**);
variable -= **exp**;	**variable** = **variable** - (**exp**);
variable *= **exp**;	**variable** = **variable** * (**exp**);
variable /= **exp**;	**variable** = **variable** / (**exp**);
variable %= **exp**;	**variable** = **variable** % (**exp**);

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

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

number *= 2 + 1;            // number = number * (2 + 1);
```

`for` loop is **NOT** a method

- The `for` loop is a ***control structure***—a syntactic structure that *controls* the execution of other statements.

- Example:
  - "Shampoo hair. Rinse. **Repeat**."

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);
```

- "I want to print a line for each number from 1 to 6"

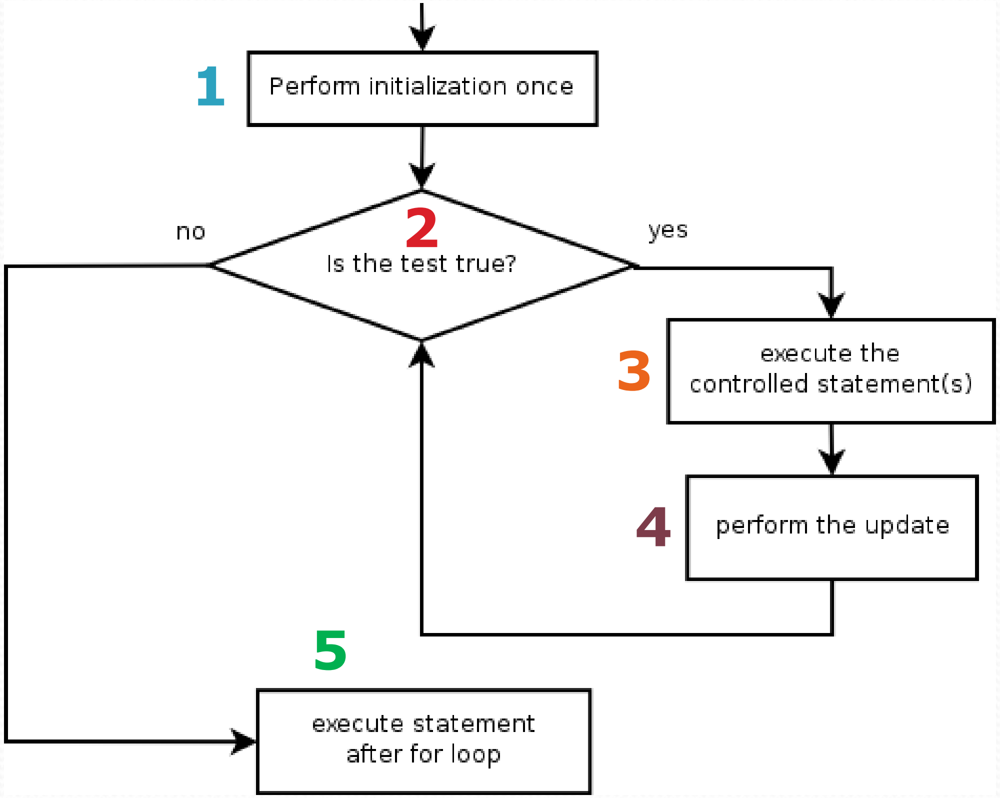
- The `for` loop does exactly that!

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

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

Loop walkthrough

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



1 Perform initialization once

2 Is the test true?

no

yes

3 execute the controlled statement(s)

4 perform the update

5 execute statement after for loop

Output:

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

Multi-line loop body

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

- Output:

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

Expressions for test

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

- This computes the Fahrenheit equivalents for -3 degrees Celsius to 2 degrees Celsius.

- 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
```

    - Concatenate `"  "` to separate the numbers

Counting down

- The **update** can use `--` to make the loop count down.
  - The **test** must say > instead of <

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

  - Output:

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