Lecture 5: For Loops

Building Java Programs: A Back to Basics Approach
by Stuart Reges and Marty Stepp

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

shortcuts to modify a variable's value

Shorthand	Equivalent longer version
variable `+=` **value**;	**variable** `=` **variable** `+` **value**;
variable `-=` **value**;	**variable** `=` **variable** `-` **value**;
variable `*=` **value**;	**variable** `=` **variable** `*` **value**;
variable `/=` **value**;	**variable** `=` **variable** `/` **value**;
variable `%=` **value**;	**variable** `=` **variable** `%` **value**;

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

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

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

The `for` loop

Repetition with `for` loops

- So far, repeating a statement is redundant:

```
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 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 <= 6; i++) {
    System.out.println("I am so smart");
}
```

- Tells Java what variable to use in the loop
  - Performed once as the loop begins
  - The variable is called a *loop counter*
    - can use any name, not just `i`
    - can start at any value, not just `1`

Test

```
for (int i = 1; i <= 6; 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

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"

- The `for` loop does exactly that!

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

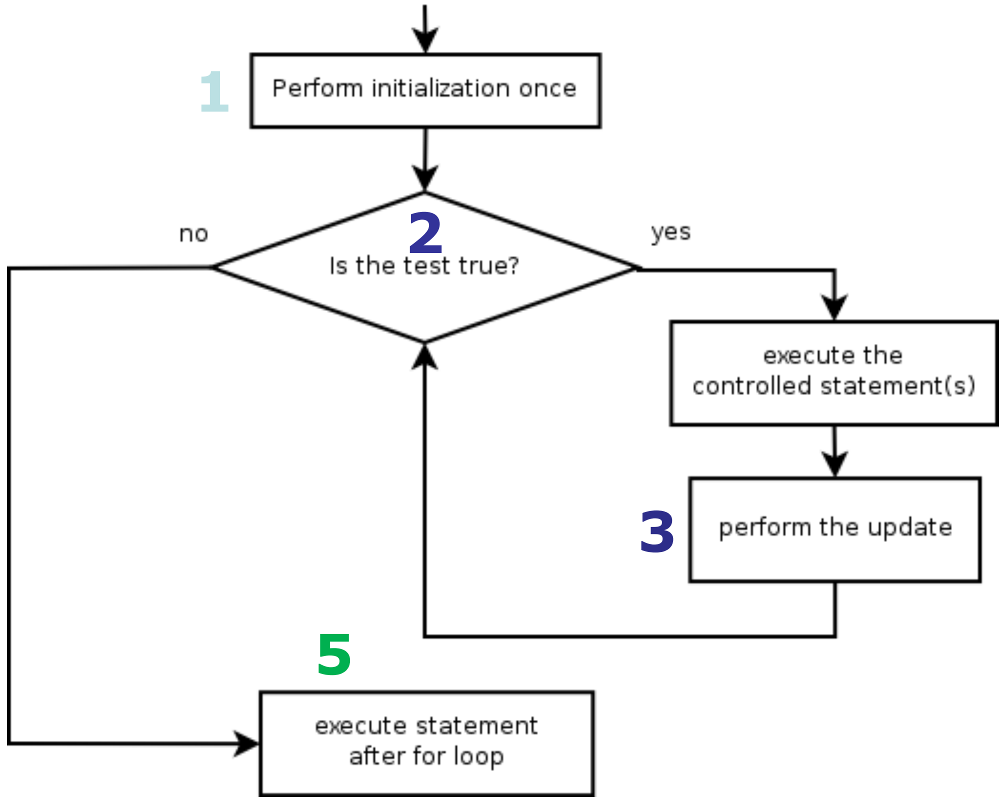
- "For each integer **i** from 1 through 6, print ..."

Loop walkthrough

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

Output:

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



1
Perform initialization once
2
Is the test true?
no
yes
execute the controlled statement(s)
3
perform the update
5
execute statement after for loop

For Loop in Movies

Groundhog Day(1993); Bill Murray.

Looper(2010); Bruce Willis and Joseph Gordon-Levitt, Emily Blunt.

Edge of Tomorrow(2014); Tom Cruise, Emily Blunt.

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
```

isPrime

```
public static boolean isPrime(int n) {
    int factors = 0;
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {
            factors++;
        }
    }

    if (factors == 2) {
        return true;
    } else {
        return false;
    }
}
```

- Calls to methods returning `boolean` can be used as tests:

```
if (isPrime(57)) {
    ...
}
```

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.
```

Example

- Write a for loop to print: 5 9 13 17 21 25

```
for (int i = 5; i <= 25; i+=4)
        System.out.print(i + ″ ‶);
  }
```

Class constants and scope

Limitations of variables

- Idea: Make a variable to represent the size.
  - Use the variable's value in the methods.

- Problem: A variable in one method can't be seen in others.

```
public static void main(String[] args) {
    int size = 4;
    topHalf();
    printBottom();
}

public static void topHalf() {
    for (int i = 1; i <= size; i++) {    // ERROR: size not found
        ...
    }
}

public static void bottomHalf() {
    for (int i = size; i >= 1; i--) {    // ERROR: size not found
        ...
    }
}
```

Scope

- **scope**: The part of a program where a variable exists.
  - From its declaration to the end of the `{ }` braces
    - A variable declared in a `for` loop exists only in that loop.
    - A variable declared in a method exists only in that method.

```
public static void example() {
    int x = 3;
    for (int i = 1; i <= 10; i++) {
        System.out.println(x);
    }
    // i no longer exists here
} // x ceases to exist here
```

i's scope

x's scope

Scope implications

- Variables without overlapping scope can have same name.

```
for (int i = 1; i <= 100; i++) {
    System.out.print(“A");
}
for (int i = 1; i <= 100; i++) {    // OK
    System.out.print(”BB");
}
int i = 5;                          // OK: outside of loop's scope
```

- A variable can't be declared twice or used out of its scope.

```
for (int i = 1; i <= 100 * line; i++) {
    int i = 2;                      // ERROR: overlapping scope
    System.out.print("/");
}
i = 4;                              // ERROR: outside scope
```

If Block

```
if(x<=3)
{
  int y=2;
  …
}

y=5; // error since y does not exist outside
     // the if block
```

Cumulative algorithms

Adding many numbers

- How would you find the sum of all integers from 1-1000?

```
// This may require a lot of typing
int sum = 1 + 2 + 3 + 4 + ... ;
System.out.println("The sum is " + sum);
```

- What if we want the sum from 1 - 1,000,000? Or the sum up to any maximum?
  - How can we generalize the above code?

Cumulative sum loop

```
int sum = 0;
for (int i = 1; i <= 1000; i++) {
    sum = sum + i; // or sum+=i;
}
System.out.println("The sum is " + sum);
```

- **cumulative sum**: A variable that keeps a sum in progress and is updated repeatedly until summing is finished.
  - The `sum` in the above code is an attempt at a cumulative sum.
  - Cumulative sum variables must be declared *outside* the loops that update them, so that they will still exist after the loop.

Cumulative product

- This cumulative idea can be used with other operators:

```
int product = 1;
for (int i = 1; i <= 20; i++) {
    product = product * 2;
}
System.out.println("2 ^ 20 = " + product);
```

  - How would we make the base and exponent adjustable?

CodingBat

CodingBat will be used to do some graded assignments. Go to the CodingBat Java at http://codingbat.com/java

1)Create an account.

2)Go to “prefs”(upper right hand, next to your login email)

3)Enter my email lnguyen8@bostonpublicschools.org under Teacher Share and click on “Share”. This will give me access to see your completed work.

4)Go to “pref” and under your name: add S1 or S2 according to your section in front of your name.

For example: S1: Nguyen, Long

Lab

Save your file in the ForLoop folder in CS50.

```
public static void main(String[] args)
{
       multipleOf3(7,18); // 9 12 15 18
       multiple4Not5(21); // 4 8 12 16
       printList(2,7); // 2 4 6
       printList(9,3);  // 3 5 7 9   }
//print multiples of 3 from m to n inclusive.
// MUST USE %
public static void multipleOf3(int m, int n){…}

// print multiples of 4 and not of 5 from 4
// to m inclusive. Assume m>=4 Must use %
public static void multiple4Not5(int m){…}
// see next page, Must use %
public static void printList(int a, int b){}
```

Lab

Write a method printList that accepts two integer as arguments. You can assume that these integers are different. If the first argument is smaller than the second, print the even integers in natural order between the arguments including, if necessary, the endpoints. If the first argument is larger than the second, print the odd numbers between them. Write out the complete method header.

The following calls from the main method should have the output below.

```
printList(2,7); // 2 4 6
printList(9,3);  // 3 5 7 9
```