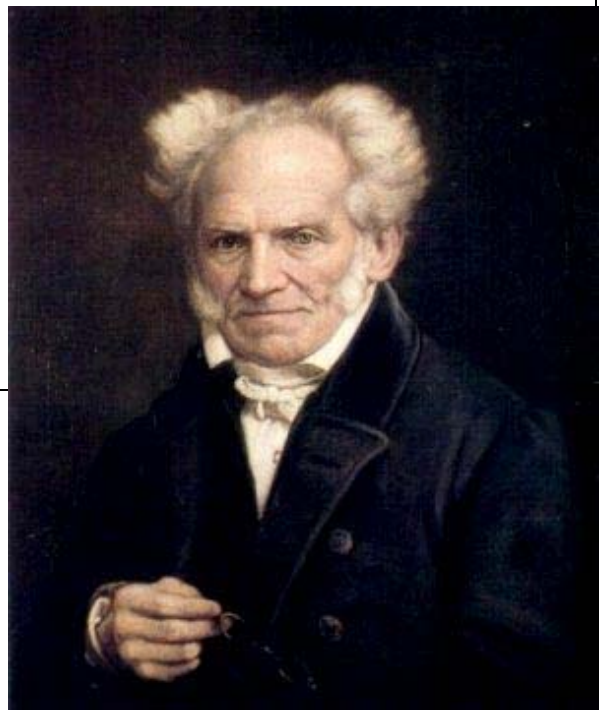
Topic 5
for Loops

"Always to see the general in the particular is the very foundation of genius."
-Arthur Schopenhauer



Based on slides for Building Java Programs by Reges/Stepp, found at http://faculty.washington.edu/stepp/book/

What we will do today

- Explain and look at the syntax and examples of
  - –for loops

`System.out.print` command

- `System.out.println` prints a line of output and then advances to a new line.
- Java has another command named `System.out.print` that prints the given output without moving to the next line.
  - This allows you to print partial messages that can appear on the same line as each other.
- Example:

```
System.out.print("Olivia and");
System.out.print("Isabelle are");
System.out.println("my daughters.");
System.out.print("I am Kelly's husband.");
```

Output:

```
Olivia andIsablle aremy daughters.
I am Kelly's husband.
```

Repetition with `for` loops

- So far, when we wanted to perform a task multiple times, we have written redundant code:

```
System.out.println("CS305J");
System.out.println();  // print 5 blank lines
System.out.println();
System.out.println();
System.out.println();
System.out.println();
System.out.println("Introduction to Computing");
```

- Java has a statement called a *for loop statement* that instructs the computer to perform a task many times:

```
System.out.println("CS305J");
for (int i = 1; i <= 5; i++) {
    System.out.println();
}
System.out.println("Introduction to Computing");
```

The `for` loop

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

- General syntax:

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

  - The top line is usually called the loop *header*, and the statement(s) inside are called the loop *body*.
  - The ***<initialization>*** usually declares a *loop counter* variable that is used in the test, update, and body of the loop.

- Semantics (behavior) of `for` loop:
  - Start out by performing the ***<initialization>*** once.
  - if the ***<test>*** is a **true** statement execute all the statements in the body of the loop in the order they appear, one time
  - After each time the loop body is executed, perform the ***<update>*** and then check the ***<test>*** again.
  - if the ***<test>*** is a false skip to the first statement **after** the body of the loop

Example `for` loop of `int`s

- The simplest way to use a `for` loop is to loop over integers in a given range:

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

- Example:

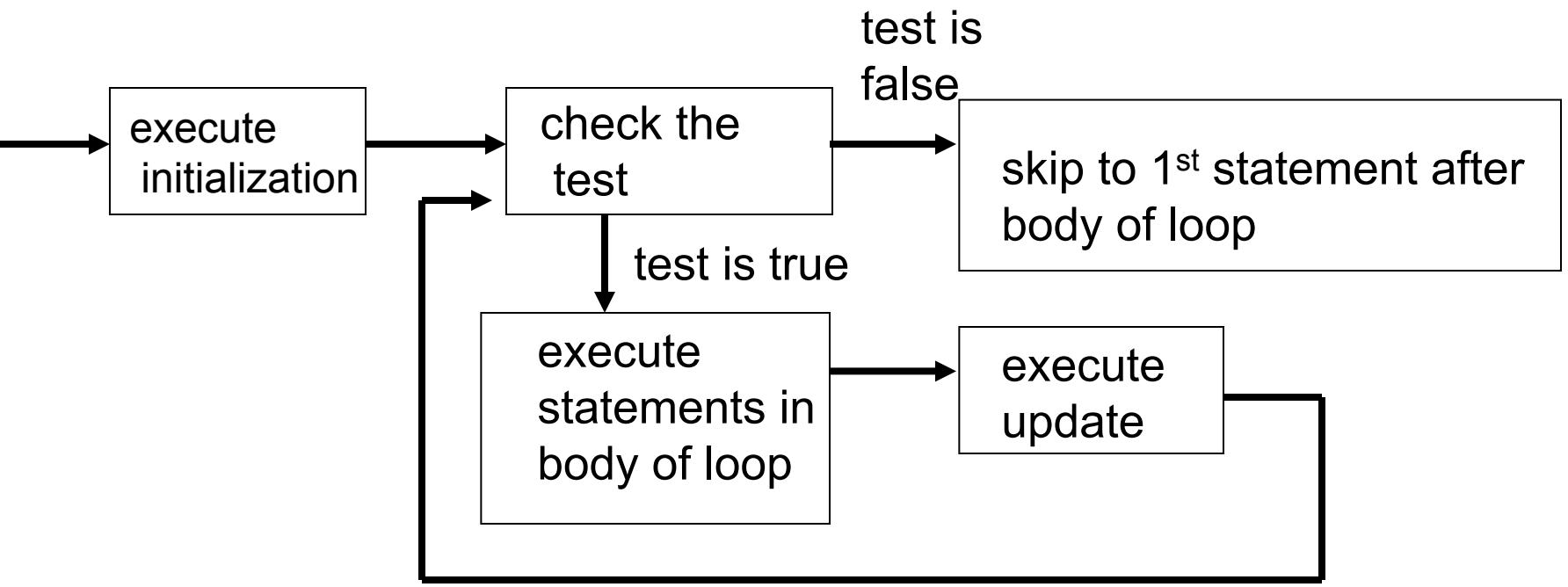
```
for (int i = 1; i <= 6; i++) {
    System.out.println(i + " squared is " + (i * i));
}
```

  - 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

- The following flow diagram describes the execution of a `for` loop:



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));
}
```

- `int i = 1;`
- is `1 <= 3?` Yes, so execute the body and update.
  ```
  System.out.println(1 + " squared is " + (1 * 1));
  i++;  // i = 2
  ```
- is `2 <= 3?` Yes, so execute the body and update.
  ```
  System.out.println(2 + " squared is " + (2 * 2));
  i++;  // i = 3
  ```
- is `3 <= 3?` Yes, so execute the body and update.
  ```
  System.out.println(3 + " squared is " + (3 * 3));
  i++;  // i = 4
  ```
- is `4 <= 3?` No, so exit the loop.

Another example `for` loop

- Example:

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

– Output:

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

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?
```

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
```

If the loop counter `i` was initialized to `1` and the test was changed to `i <= 6`, how would you change the loop body to get the same output show to the left?

Complex Example

Complex Example:

```
for (int i = 1 + 3 * 4; i <= 5298 % 100; i++)
  System.out.println(i + " squared is " + (i * i));
```

The above for loops is syntactically correct but it is difficult to understand.

Strive to make you for loops, like your code, **easy to understand.**

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 = 5; i >= 1; i--) {
     System.out.print(i + " ");
}
System.out.println("Blastoff!");
```

Output:

```
T-minus 5 4 3 2 1 Blastoff!
```

Rewrite this using an upward counting loop

Degenerate loops

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

```
// The 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. These are called *infinite loops*.

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

for Loop Exercises

- Write a for loop to produce the following output

```
30, 20, 10, 0, -10, -20
-7, -3, 1, 5, 9, 13
```

A single line of output with 200 `!`'s

1,000,000 `!`'s, with 5 `!`'s per line.

- Loops = power!

useful for loop example

- Heron's method for calculating square roots
- problem: Find sqrt(n)
- Algorithm:

1. Make a guess at the solution. (x_1)
2. $x_2 = (x_1 + (n / x_1)) / 2$
3. Repeat for $x_3, x_4, x_5, \ldots$

Write a Java program that implements Heron's method to find the square root of 133,579 using 20 iterations of the algorithm.