Kaitlin Hoffmann

Office Hours:

SH 243 MR 11:00 - 12:30 PM via appointment https://calendly.com/hoffmank4/15min

Email: hoffmank4@newpaltz.edu

For TA Office Hours and Email – Please see syllabus

NEW! Supplemental Instruction: Sign up at my.newpaltz.edu

FOR LOOPS

COMPUTER SCIENCE I

OBJECTIVES

- Review
- for Loops

- What will the following produce in Java?

1.
```java
int num = 45;

if(num <= 10 && num > 1) {
    System.out.println("Hello");
} else if(num <= 50) {
    System.out.println("Goodbye");
} else {
    System.out.println("This is a pointless program.");
}
```

1.

```java
int num = 45;

if(num <= 10 && num > 1) {
    System.out.println("Hello");
} else if(num <= 50) {
    System.out.println("Goodbye");
} else {
    System.out.println("This is a pointless program.");
}
```

Goodbye

- What will the following produce in Java?

2.

```
int num = 45;

if(num <= 10 && num > 1) {
    System.out.println("Hello");
}
if(num <= 50) {
    System.out.println("Goodbye");
}
if(num == 45) {
    System.out.println("Wow");
}
```

2.

```java
int num = 45;

if(num <= 10 && num > 1) {
    System.out.println("Hello");
}
if(num <= 50) {
    System.out.println("Goodbye");
}
if(num == 45) {
    System.out.println("Wow");
}
```

Goodbye

Wow

LOOPS

- A **loop** can be used to tell a program to execute statements repeatedly.
- Suppose that you need to display a string (e.g., **Welcome to Java!**) a hundred times. It would be tedious to have to write the following statement a hundred times:

```
100 times { System.out.println("Welcome to Java!");
            System.out.println("Welcome to Java!");
            ...
            System.out.println("Welcome to Java!");
```

LOOPS

- Java provides a powerful construct called a loop that controls how many times an operation or a sequence of operations is performed in succession.
- Using a loop statement, you simply tell the computer to display a string a hundred times **without** having to code the print statement a hundred times, as follows:

```
for(int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```

FOR LOOPS

```
for(int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```

- The above is an example of a **for** loop. The parts it's made up of:

1. **initializing statement**: int i = 0
2. **testing condition**: i < 100
3. **increment/decrement**: i++
4. **Code to be iterated/repeated:**

   System.out.println("Welcome to Java!");

- Lets talk about each part…

INITIALIZING STATEMENT

```
for(int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```

- The **initializing statement** marks the *beginning* of the loop structure.
- It contains a variable with some **initial value** that is defined by the programmer.
- This is the value of the control variable when the control shifts into the loop. However this statement is **executed only once**.
- In programming, it's typical to start at 0 not 1. We will see one reason why when we learn about arrays later on in the course. For a more extensive explanation, see the answers provided here (optional): https://www.quora.com/Why-do-we-count-from-0-in-most-programming-languages

TESTING CONDITION

```
for(int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```

- The **testing condition** (a.k.a. the loop-continuation-condition) is a Boolean expression that **controls** the execution of the body.
- As long as the testing condition is **true**, the body is executed. Once it's false, the iteration is stopped, and the program moves on to the next line of your program.
- To explain this simply, the example above is stating:
  - ✦ while i is less than 100, print "Welcome to Java!".
- **NOTE:** The for loop is printing "Welcome to Java" 100 times, but since we are starting i at 0, our testing condition is i < 100, **NOT** i <= 100.

TESTING CONDITION

```
for(int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```

- **NOTE:** The for loop is printing “Welcome to Java” 100 times, but since we are starting i at 0, our testing condition is i < 100, **NOT** i <= 100.
- Example:

```
int amountLoopRan = 0;
for(int i = 0; i < 5; i++) {
    System.out.println("i = " + i);
    amountLoopRan++;
}

System.out.println("Loop ran " + amountLoopRan + " times.");
```

Output

```
i = 0
i = 1
i = 2
i = 3
i = 4
Loop ran 5 times.
```

INCREMENT/DECREMENT

```
for(int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```

- The **increment/decrement** (the action-after-each-iteration) is controls the value of i after iteration.
- In the example above, i is **increased** by 1 after every iteration.
- i doesn't have to be just increased or decreased by just 1; You can use any value necessary. You also don't have to use i as the variable name. You can name the initialization statement whatever you want. **Example:**

```
for(int num = 0; num < 10; num+=2) {
    if(num % 2 == 0) {
        System.out.println(num);
    }
}
```

Output

```
0
2
4
6
8
```

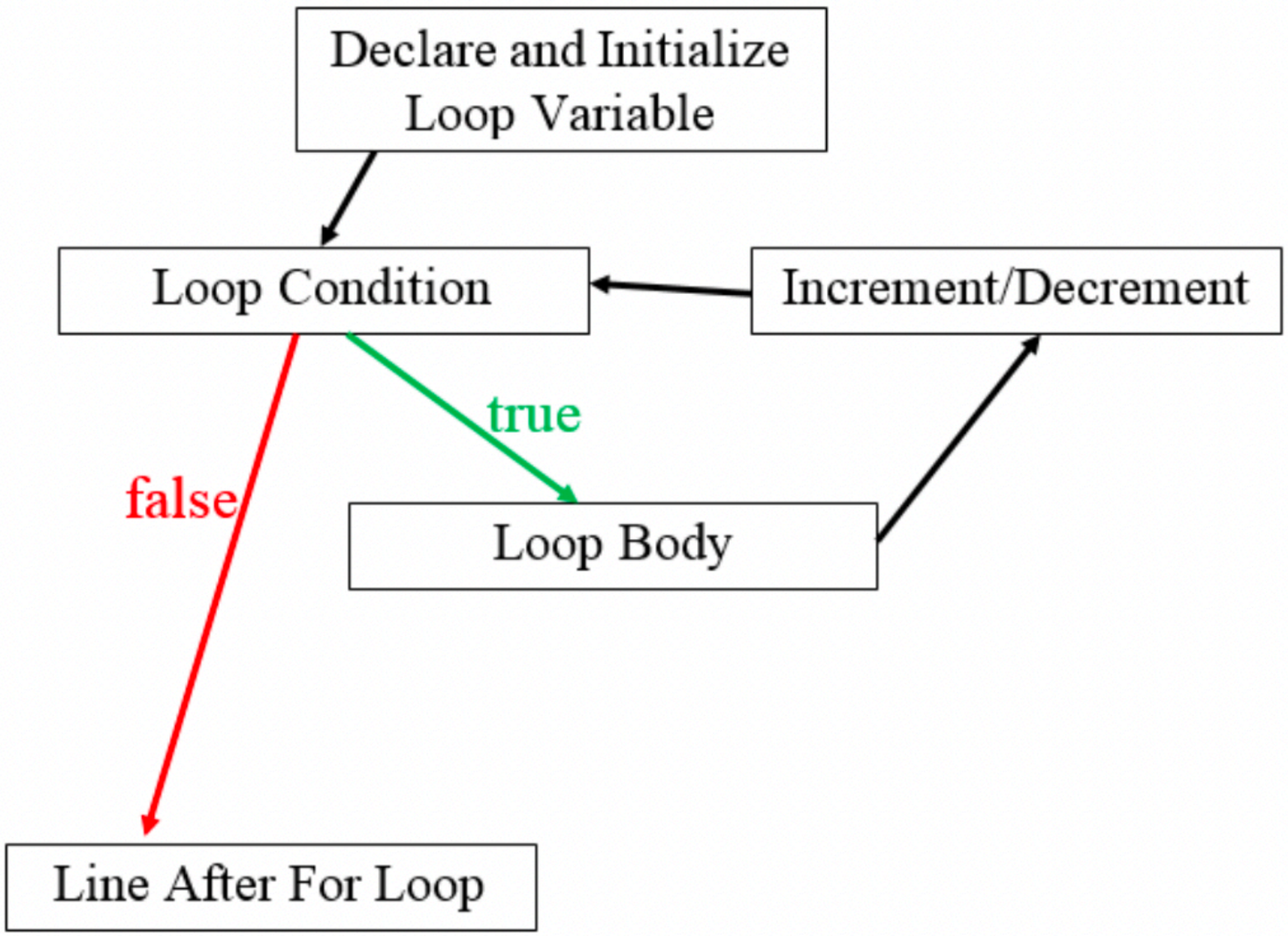
CODE TO BE ITERATED

```
for(int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```

- Inside the for loop body is where code is written to be **iterated**.
- In the simple example above, we are just printing. However, like we saw before, you may want to write conditional statements, calculate some kind of equation, etc.
- You can also use loops to check user input with Scanner. However, while loops are used for this normally which we will see soon.

FOR LOOP FLOW

```
for(int i = 0; i < 100; i++) {
    System.out.println("Welcome to Java!");
}
```



EXAMPLE 1

- Print 1 through 10 on the same line using a for loop. Your initialization statement should start at 0. (We should NEVER hard code these numbers (ie. 1 through 10)).

EXAMPLE 1

- Print 1 through 10 on the same line using a for loop. Your initialization statement should start at 0. (We should NEVER hard code these numbers (ie. 1 through 10)).

```
public class Main {

    public static void main(String[] args) {

        for(int i = 0; i <= 10; i++) {
            if(i > 0) {
                System.out.print(i + " ");
            }
        }

    }
}
```

Output

```
1 2 3 4 5 6 7 8 9 10
```

EXAMPLE 2

- Find the sum of all the integers between 1 and 10 inclusive. Display the sum.

EXAMPLE 2

- Find the sum of all the integers between 1 and 10 inclusive. Display the sum.

```java
public class Main {

    public static void main(String[] args) {

        int sum = 0;
        for(int i = 0; i <= 10; i++) {
            sum+=i;
        }
        System.out.println("sum = " + sum);

    }
}
```

Output

```
sum = 55
```

EXAMPLE 3

- Print the positive numbers that are odd and less than 50.

EXAMPLE 3

- Print the positive numbers that are odd and less than 50.

```
//Two ways to solve:
 //can solve using a condition
for(int i = 0; i < 50; i++) {
    if(i % 2 == 1) {
        System.out.print(i + " ");
    }
}

 System.out.println("\n");

//or solve by starting at 1 and increasing by 2
for(int i = 1; i < 50; i+=2) {
    System.out.print(i + " ");
}
```

Output

```
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49
```

YOUR TURN — EXERCISE 1

- Print 5 to -5 using a for loop.

YOUR TURN — EXERCISE 1

- Print 5 to -5 using a for loop.

```
public class Main {

    public static void main(String[] args) {

        for(int i = 5; i >= -5; i--) {
            System.out.print(i + "  ");
        }

    }
}
```

Output

```
5  4  3  2  1  0  -1  -2  -3  -4  -5
```

YOUR TURN — EXERCISE 2

- Print the even numbers between 25 and 100 using a for loop.

YOUR TURN — EXERCISE 2

- Print the even numbers between 25 and 100 using a for loop.

```
//two ways to solve:

//first way
for(int i = 25; i <= 100; i++) {
    if(i % 2 == 0) {
        System.out.print(i + " ");
    }
}

System.out.println("\n");

//second way
for(int i = 26; i <= 100; i+=2) {
    System.out.print(i + " ");
}
```

Output (goes to 100...)

```
26 28 30 32 34 36 38 40 42 44 46 48 50 5
26 28 30 32 34 36 38 40 42 44 46 48 50 5
```