Dr. Nazli Hardy, MBA, Ph.D.

for Loops

Adapted from "Building Java Programs" Reges & Stepp

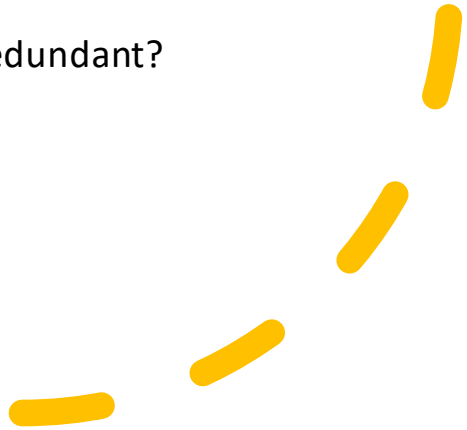
Necessary Tools: Paper & Pencil

Repetitions with *for* loops

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

```
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("Really?");
System.out.println("Yup, I am all that!");
```

What have we already used to make code more efficient/ less redundant?



Efficient Repetitions with *for* Loops

- Happily, programming allows us to create a ***for loop statement*** that instructs the computer to perform a task many times.

```
for (int i = 1; i <= 5; i++){
// repeat 5 times
    System.out.println("I am so smart");
}

System.out.println("Really?");
System.out.println("Yup, I am all that!");
```

Chap2/LoopEx0

for Loop Syntax

`for` loop: a statement that executes a group of statements repeatedly until a given test fails

General syntax:

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

Example:

```
for (int i = 1; i <= 10; i++) {
  System.out.println("Her name is Kathryn Stockton");
  System.out.println("She is the author of 'The Help'");
}
```

Chap2/LoopEx1

Thank Heavens for *for* Loops

Imagine having to print out the following 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
7 squared is 49
8 squared is 64
9 squared is 81
10 squared is 100

Or worse … up till 1346 squared

for Loop over Range of Integers

We'll write *for* loops over integers in a given range.

Example:

```
for (int i = 1; i <= 10; i++) {// repeat 10 times

    System.out.println(i + " squared is " + (i * i));
}
```

"For each int **i** from 1 through 10, ..."

Output:

```
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
:
10 squared is 100
```

Notice declaration of variable AND use of concatenation

Chap2/LoopEx4

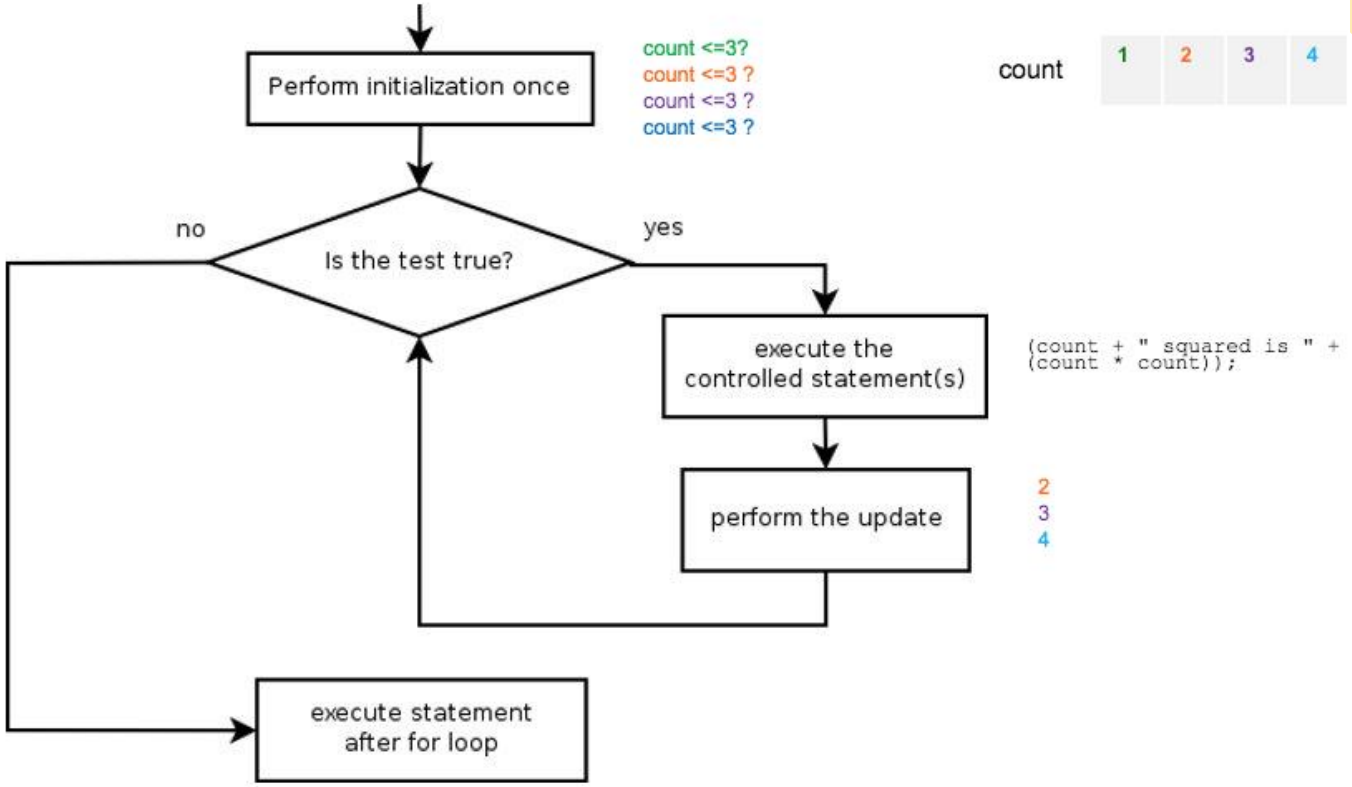
for loop flow diagram and loop walkthrough

Let's walk through the following **for** loop:

```
for (int count = 1; count <= 3; count++) {
    System.out.println(count + " squared is " + (count * count));
}
```

Output

```
1 squared is 1
2 squared is 4
3 squared is 9
```



Chap2/LoopEx5

Example *for* Loops

Chap2/LoopEx5B

```
public static void main(String[] args) {

 for (int count = 1; count <= 10; count++) {

   System.out.print(count + " ");

 }


 System.out.println();

 for (int count = 0; count < 10; count++) {

   System.out.print(count + " ");
 }


 System.out.println();

 for (int count = 10; count >= 0; count- -) {

   System.out.print(count + " ");

 }

}
```

Output?

- The body of a `for` loop can contain multiple lines.

Example 1:

```
System.out.println("+----+");

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

System.out.println("+----+");
```

Output:

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

Notice use of curly brackets

This code looks almost the same but what difference do you notice?

Example 2:

```
System.out.println("+----+");

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


System.out.println("+----+");
```

Output?

Chap2LoopEx6

More *for* Loops. [Use sparingly]

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
```

Example:

```
for (int i = 1 + 3 * 4; i <= 5248 % 100; i++) {
    System.out.println(i + " squared is " + (i * i));
}
```

What is the initialization value?
What is the test value?

Chap/LoopEx7

Downward Counting *for* Loops

Remember that update can also be a `--` to make the loop count down, instead of up
Notice use of print and println

This also requires changing the test to e.g. `>=` instead of `<=`

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

Output:

Chap/LoopBlastOff

Note: what do you notice about the use of curly brackets here?

Increments & Decrements

i++ i+=1

i-- i-=1

i+=2 i+=3 i+=4 ....

i-=2 i-=3 i-=4 ....

i*=2 i*=3 i*=4 ....

i/=2 i/=3 i/=4 ....

Critical Thinking *

Write a loop that produces the following output.

```
On day #1 of Christmas, my true love sent to me
On day #2 of Christmas, my true love sent to me
On day #3 of Christmas, my true love sent to me
On day #4 of Christmas, my true love sent to me
On day #5 of Christmas, my true love sent to me
...
On day #12 of Christmas, my true love sent to me
```

```
public static void main(String[] args) {

for (_____ ) {


System.out.println(" _____ ");

}
```

Questions to consider:

- What is the pattern here?
- How many times do we need to repeat something?
- What is being repeated?
- And note use of concatenation

Chap2/LoopEx9

Checking In

Mapping for Loops to Numbers

for loop to produce the following output?

```
2 4 6 8
Who do we appreciate?
```

Let's consider 2 ways to do this:

1st way with an update i+=2

```
public static void main(String[] args) {

for (_____ ) {

   System.out.print(" _____");

}

System.out.print("\nwho do we appreciate?");

}
```

Questions to consider:

- What is the pattern here?
- How many times do we need to repeat something?
- What is being repeated?
- And note use of concatenation

Chap2/LoopEx10

Mapping for Loops to Numbers* (2nd way – better way)

IMPORTANT – relating back to i

for loop to produce the following output?

```
2  4  6  8
Who do we appreciate?
```

Hint: Look for patterns – especially related to the variable

Draw a table – and follow my example

```
for (_____ ) {

   System.out.print(“ _____”);

}

System.out.print("\nwho do we appreciate?”);
```

Questions to consider:

- What is the pattern here?
- How many times do we need t repeat something?
- What is being repeated?

i	output	relation to i
1	2	2 * i
2	4	2 * i
3	6	2 * i
4	8	2 * i

Chap2/LoopEx10b

Mapping for Loops to Numbers *

Questions to consider:

What is the pattern here?

How many times do we need to repeat something?
What is being repeated?

for loop to produce the following output? **Use the best way**

```
3  6  9  12  15
```

Hint: Look for patterns – especially related to the variable
Draw a table

i	output	relation to i
1	3	3 * i
2	6	
3	9	
4	12	
5	15	

```
public static void main(String[] args) {

   for (_____ ) {


      System.out.print _____ ");
   }
```

Chap2/LoopEx11

Mapping for Loops * (Group Exercise)

for loop to produce the following output?

```
4  7  10  13  16
```

i	output	relation to i
1	4	
2	7	
3	10	
4	13	
5	16	

```
public static void main(String[] args) {
    for (_____ ) {

      System.out.print( _____ );

   }
```

Chap2/LoopEx12

*Mapping `for` Loop To Numbers * (Class Exercise)*

- What statement could we write in the body of the loop that would make the loop print the following output?

```
2  7  12  17  22
```

Chap2/LoopEx14

Patterns to notice:

Increment?

What number is a common thread?

- To find the pattern, it can help to make a table of the count and the number to print.

i	output (number to print)	relation to i
1	2	
2	7	
3	12	
4	17	
5	22	

```
for (_____ ) {

   System.out.print( _____ );

}
```

Mapping for Loops * (Group Work)

What statement could we write in the body of the loop that would make the loop print the following output? 2 good ways ..

```
17 13 9 5 1
```

count	output (number to print)	relation to count
1	17	21-(4*i)
2	13	
3	9	
4	5	
5	1	

```
for (_____ ) {

System.out.print( _____ );

}
```

count	output (number to print)	relation to count
5	17	(4 * i) - 3
4	13	
3	9	
2	5	
1	1	

```
for (_____ ) {

System.out.print _____ ");

}
```

Chap2/LoopEx15

Constants

Recall: Variables

Variable declaration

```
int size = 5;
double range = 7. ;
```

Constants

Constant declaration

```
public static final int SIZE = 7;
public static final double RANGE= 7. ;
```

Constant Example *

```
public class Prelab4Constant1 {

public static final int LENGTH = 7;

public static void main(String[] args) {

drawLine();

}


public static void drawLine() {

// draw a cross at the beginning
System.out.print("+");

// draw =*
for (int i = 1; i <= LENGTH*2; i++) {
System.out.print("A");
}

// draw a cross at the end
System.out.println("+");

}
}
```

Practice, Practice, Practice
Pre Lab 4 Exercise 1

Note on Upcoming Lab

Create new Java Project: LoopPractice

Run **each and every one** of the starred examples (*) in this presentation as a separate class

[I will ask to see this as evidence of work completed]

Practice, Practice, Practice
Pre Lab 4 Exercise 2

Within your Java Project: LoopPractice

Write a for loop that creates the following pattern

+=*=*=*=*=*=*=*=*=*=*=*=*=*=*=*+

Practice, Practice, Practice
Pre Lab 4 Exercise 3

Within your Java Project: LoopPractice rewrite the *for* loop that creates the following pattern, using a **constant of 7**

+=*=*=*=*=*=*=*=*=*=*=*=*=*=*+

Let's discuss why constants are useful

Chap2/PPreLab4Constant2
Chap2/PrePreRocket