LOOPS

BY ANNEMARIE CABALLERO

I.T. GIRLS AUGUST 13-17

LOOP

- Iterative statement (repeats until a certain condition is no longer true)
- 3 parts:
  - Initialization – create the variable to be tested and give it a value
  - Testing – test the variable to see if the desired condition is true
  - Change – change the variable after it has been tested (or you'll have an infinite loop
- 3 main types:
  - For
  - For Each (will learn about in arrays)
  - While
- Why use loops?
  - Avoid repeating code
  - Example: Printing out the first 100 numbers

PROGRAMMERS COUNT FROM ZERO

FOR LOOPS

- Iterates through by checking a number
- Only need the change in parentheses
- for(initialization; test; change) { … }
- Examples:

```
//prints numbers 0 – 9
for(int i = 0; i < 10; i++) {
        System.out.println(i);
}
//variable i exists only inside
the loop
```

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

WHILE LOOP

- Remember to change the test variable (not in the top statement like in for loops)
  - As such, while loops can often turn into infinite loops

```
//prints 0-9
int num = 0;
while (num < 10) {
	System.out.println(num);
	num++;
}
```

IMPORTANT RESERVED WORDS

- Return
  - Will break you out of a loop
  - Example: Return the first composite number
- Break
  - Will exit the loop and go to first code below it
  - Should always be used inside an if
  - Example: Prints prime numbers until it reaches a composite
- Continue
  - Will proceed to the next iteration of loop without executing remaining code below it
  - For will go to the update statement, While will go to the condition
  - Example: Prints only prime numbers

NESTED LOOP

putting loops
inside other loops

```
*
**
***
****
… until n rows of stars
```

```
for (int row = 1; row <= n; row++) {
        for(int count = 1; count <= row; count++)
                System.out.print("*");
        System.out.println();
}
```