CS 171G
Lab 5 – Programming with If Statements and Loop

Conditionals and Loops

In the Java programs you've seen so far, we've only performed statements in sequence, meaning step-by-step, in order.

Now we will examine programming statements that allow us to:

- make decisions (conditionals)
- repeat processing steps (loops)

Flow of Control

Unless specified otherwise, statements are executed one statement after another, in sequence. Some programming statements allow us to decide whether or not to perform a task. These decisions are based on boolean expressions (or conditions) that evaluate to true or false.

The order of statement execution is called the **flow of control**.

Conditional Statements

A conditional statement lets us choose which statement will be executed next. Therefore they are sometimes called selection statements. Conditional statements give us the power to make decisions.

Java allows us to make decisions with the ***if statement*** and the ***if-else statement***.

The If Statement

The if statement has the following syntax:

```
if ( condition )  {
   // perform these statements
}
```

The `condition` must be a boolean expression. It must be either true or false.

The If-else Statement

The if-else statement has the following syntax:

```
if ( condition )  {
   // perform these statements if condition is true
}
else {
  // perform these statements if condition is false
}
```

Example: If Statement

```
if (x > 3)
   System.out.println( "okay" );
```

This would print the word **okay** if the value of x is greater than 3. If x isn't greater than 3, this code wouldn't print anything.

Example: If-else Statement

```
if (x > 3)
   System.out.println( "okay" );
else
   System.out.println( "not okay");
```

This would print the word **okay** if the value of x is greater than 3. If x isn't greater than 3, the else portion of this code would print the words **not okay**.

Boolean Expressions

Many conditions use comparison operators which all return boolean results. Here are the comparison operators that you can use in Java.

==	equal to
!=	not equal to
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to

Note the difference between the equality operator == and the assignment operator =

Another example of an if-else statement

In JavaScript Module 1 there was an example if-else statement. Although JavaScript and Java are different languages, it would look very similar in Java.

```
lunchCost = 7.00;
moneyInPocket = 6.25;
if (moneyInPocket >= lunchCost) {
   System.out.println("Yeah! Let’s eat!");
}
else {
   System.out.println("I’m hungry…");
}
```

Logical Operators

Boolean expressions can also use the following logical operators:

```
!    Logical NOT
&&   Logical AND
||   Logical OR
```

They all take boolean operands and produce boolean results

Loops - The while Statement

A while statement has the following syntax:

```
while ( condition ) {
   // perform these statements if the condition is true
}
```

Example: While Loop

```
// print the powers of 2 that are less than or equal to 12
int x = 1;
while (x <= 12) {
   System.out.println( "x is " + x);
   x = x * 2;
}
```

The output of this loop will be:

```
1
2
4
8
```

Here are some uses for loops:

- A loop can be used to maintain a running sum
- A sentinel value is a special input value that represents the end of input
- A loop can also be used for input validation, making a program more robust

Infinite Loops

The body of a while loop eventually must make the condition false.

If not, it is called an **infinite loop**, which will execute forever.

This is a common logical error.

You should always double check the logic of a program to ensure that your loops will terminate normally.

Loops - The for Statement

A **for** statement has the following syntax:

```
for ( initialization ; condition ; increment )
   // perform these statements if condition is true
```

Example: For Loop

```
for (int n = 1; n < 9; n++)
    System.out.println ( n * 5 );
```

The output of this loop will be:

```
5
10
15
20
25
30
35
40
```

1. Follow the following steps to create a Java Application.

    a) Open NetBeans. Choose File / New Project. Under Categories, select Java. Under Projects, select Java Application. Click Next.

    b) Under Project Name, enter the name of your application (Ex: Lab5). Change the Project Location to any folder on your computer. Make sure that the checkbox next to Create Main Class is checked.

    c) Click Finish. The IDE creates the project folder. You will see this source code on the right, which looks like this:

```
package lab5;

/**

 *
 * @author quando
 */
public class Lab5 {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }
}
```

A Java comment begins with two forward slashes.

1. At the top of the program, add comments with your name and today's date like this:

   // CS 171G – Lab 5
   // written by ……………………….
   // today is ………………………………

   2. Just below the line that says "// TODO code application logic here", type in the while loop shown

      ```
      // step 2 - while loop
      // print the powers of 2 that are less than or equal to 12

      int x = 1;
      while (x <= 12) {
         System.out.println( "x is " + x);
         x = x * 2;
      }
      ```

- Compile (debug syntax errors if needed). Then run the program.

- Check your output. (compare to the output shown on page 3)

3. Change the first comment to

   ```
   // step 3 – while loop
   ```

   Then add a print statement to print a blank line.

   ```
   System.out.println( );
   ```

   Copy the loop and paste it below this comment.

   Change all of the occurrences of x to y.

   Change the condition of the loop so that it will print all of the powers of 2 that are less than 100.

   Compile and run the program.

   You should see the 1,2,4,8 from the first loop and then 1, 2, 4, … 64 from the second loop.

4. Add a print statement to print a blank line.

   ```
   System.out.println( );
   ```

   Add the comment and declare two variables:

   ```
   // step 4 – if else statement
   int num1 = 5, num2 = 15;
   ```

   Add an if-else statement that will print "yes" if num1 is bigger than num2 and print "no" if it is not bigger.

   Compile and run. Debug as needed.

5. Print a blank line. Add the comments and declare variables

   ```
   // step 5 – for loop
   // print the multiples of 3 from 18 to 51
   int num3 = 18;
   ```

   Add a for loop:

   ```
   for ( int p = num3; p <= 51; p = p + 3 )
        System.out.print( p + " ");
   ```

   Compile and run. Debug as needed.

6. Print a blank line. Add appropriate comments and variables. Then add a for loop that will print the multiples of 5 from 5 to 30, all on the same line.

   Compile and run. Debug as needed.

   Submit Lab5.java on Canvas.