# Java Control Flow - LAB

The following instructions are to follow the lecture on types and variables. These should be done in a tool that allows for immediate compilation as some of the instructions will be to enter something that will break. This is in order to see how the compiler enforces some of the language rules. There will also be heavy use of the the `System.out.println()` method to see the results once the code compiles.

Keep in mind the instruction *can* be written in a plan text editor and then compiled on the command line using `javac` but the process will be much slower.

As discussed in the lecture, control flow statements exists to allow the programmer to control the execution flow of a program. This is done primarily through the use of decision statements and loops. Both of those will be heavily highlighted in this section. As a part of each of those statements the final control, the branch statement, will also be discussed.

The `Math.random()` function is used to create a random double number between 0.0 and 1.0. Make that an integer value between 1 and 100 by using the following statement:

```
int randOne = (int)(Math.random() * 100);
```

Next perform the following evaluations using multiple `if` statements.

- If the number is greater than 70 print the string "Big Number".
- If the number is less than 40 print the string "Small Number".
- If the number is in between 40 and 70, or equal to 40 or 70, print the string "Medium Number".

Write the same tests as above but using an `if-else` statement. Which is more efficient? Easier to code? Easier to read? Easier to maintain?

Could the above `if` statement be written with an equivalent `switch` statement? (Short answer - yes.) What would be the answers to more efficient? Easier to code? Easier to read? Easier to maintain?

**Add Nesting**
Use complex conditionals ( that AND that ) to add another level of testing.

- If greater than 90 print "Really big number"
- If less than 15 print "Really small number"
- If equal to 50 print "Really average number"

Next use nested `if-else` statements to perform the sames tests as above.

***for loops***
Write a for loop that prints out the numbers 1-30.

Write a for loop that prints out the even numbers 1-30. How many ways can this be done? Which is more efficient?

Write a loop that counts down from 20 to 0.

Although not very common, the various parts of a `for` loop can contain multiple statements. Imagine that you want to declare two variables, x and y, and you wanted to print their values counting by 1 and by 4. Until x was greater than 20 or y was greater than 50. How would you do that? Type in the following statement and examine the output:

```
for (int x = 1, y = 1; x < 20 && y < 50; x++, y += 4)
{
  System.out.println("X is: " + x + ", y is: " + y);
}
```

Is the and operator ( `&&` ) surprising? The statement above had the phrase "or" in it, why would the statement have an "and"? Remember, in the && operator it only takes one operand to be false to make the statement false. And the `for` loop runs as long as the entire statement is true. So once either of the statements become false the loop should quit. As long as they are both true the loop should continue.

Declare a very long String of at least 10 words. Something such as:

```
String str = "This is a very long and completely meaningless string that serves
no purpose.";
```

Write a statement using an enhanced `for` that will print out each character on its own line. Note that the enhanced `for` will not work on a String so you will have to consult the documentation on how to get an iterable collection of characters from a string.

**Switch statement**
Use the looping structure from above and alter the process just a bit. For each vowel, print the capital form of it instead of the lower case. Try writing the logic with both an `if` statement and then a `switch` statement. Which is easier?

**while**
Use the premise of the random number between 1 and 100 mentioned earlier in the lab, but intialize the value to 0. Write a `while` loop that will print out the value and then generate a new value. The loop should end if either the generated value is greater than 85 or the loop has been executed 20 times.

**do while**
Use the premise of the random number between 1 and 100 again, this time to initialize two different variables. Using a `do-while` loop, calculate and print the value of the two numbers multiplied together and exit the loop if the result is more that 1000 or the loop has been executed 30 times.

**using break, continue, and return** Earlier you wrote a `for` loop that printed even numbers. One way was to start at 0 and write the increment expression as `i+=2`. Write the loop again but use the modulo operator and the `continue` statement to only print even numbers.

Write nested `for` loops that will print out the sum of the inner loop up to the index of the outer loop. For example, the first time through the output should be similar to: `outer: 1, inner sum: 1`, the second time through the output should be `outer: 2, inner sum: 3`, etc. Do this for an outer loop of 1 to 30.

Modify the behavior so that if the inner sum is greater than 350, the outer loop will print the index where this threshold was reach and then quit without executing any more iterations.

Earlier there was a scheme to generate a random number between 1 and 100. To generalize this create a function called `makeBoundedRandom` that takes a single integer input and returns an integer. This function should do the follow:

- check that the input is greater than 10. If it is less than ten the function should alert the caller to an error by producing a result of -1. If the input is greater than 10 the function should produce a random integer between 1 and that number.

Next write a for loop to generate 20 random numbers with a limit of 1000 and print each one.

Write a loop that generates random numbers, each with a different range of 50 up to 100 and print out each one.