**Iterations Lab One**

**Document Id 2**

In this section we worked through some introductory material for Iterations with Java Loops and counting conditions within loops to express complex patterns.

**Questions**

1 ) The diference between Recursion and Iteration in Java is

1. Java does not implement Recursion
2. Any recursive code can be written as an iteration, its safer Java code to do this
3. Nested loops are harder to count, recursion is cleaner so it is better
4. Counting loops use more memory than Recursion
5. None of the above

2 ) Variables not in the lexical scope of a Lambda function must be

1. Initialised to a non null value
2. Final variables
3. Effectivley Final variables
4. Volatile variables
5. Immutable variables

**Lab Practical Task**

3.) We saw how to easily implement some tricky counting conditions with the range function for Java in Java 8 streams. We learned how to use streams and lambda functions to simplify conditional counting.

We saw how the code below

**int** j = 10;

**for** ( ; j >= 0; j--) {

**int** finalJ = j;

IntStream.*range*(0, 10 -j ).forEach(s -> {

System.***out***.print(**" "**);

});

IntStream.*range*(0, 2\*j-1).forEach(s -> {

System.***out***.print(**"\*"**);

**if**(s == 2\*finalJ -2) {

System.***out***.println();

}});

}

Produces the following output (upside pyramid )

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\***

**\*\*\*\*\***

**\*\*\***

**\***

using the above code as a guide produce an iterative loop that produces the following output

( a vertical pyramid)

**\***

**\*\*\***

**\*\*\*\*\***

**\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***