Seneca College **Mar 16, 2020**

Applied Arts & Technology
School of Computer Studies

JAC444 Final Code Submission: Apr 10, 2020

Workshop 10

Notes:

- **i.** Make sure you have all security and check measures in place (with proper use of Exceptional Handling wherever needed), like wrong data types etc.
- **ii.** Make your project in proper hierarchy; introduce proper class coherence in your project. Proper packages and **your project should be handled by only one main method which should be in a TesterClass**.
- **iii.** Given output structure is just for student to have a glimpse what the output can look, students are free to make the output better in any way.
- **iv.** The final code as zip should be submitted by the midnight to avoid late penalties which are 10% each day late.
- **v.** The workshop will be marked as follows:
  - **a.** Proper naming of the class/s expected.
  - **b.** Proper documentation for all the class/s, method/s etc. used.
  - **c.** Clear naming for the variables, class/s, methods expected.
  - **d.** Output should be clear, and sentences should make sense.
  - **e.** Clear all debugging fields, data, line etc. used in the code.

Other inputs can be given during demo, so make sure you test your program properly.

Design your own outputs for both the tasks

Task 1:

Describes how to perform matrix addition. Suppose you have multiple processors, so you can speed up the matrix addition. Implement the following method in parallel.

```
public static double[][] parallelAddMatrix(double[][] a, double[][] b)
```

Write a test program that measures the execution time for adding two 2,000 * 2,000 matrices using the parallel method (by running the multiple threads) and sequential method (calling normal methods one by one), respectively.
Hint: Divide your matrix in two 4 matrixes then run the addition of all in 4 threads and at the end join the threads to finish he process one after another and calculate the time of all the threads.

Task 2:

Write a program called ReverseThread.java that creates a thread (let's call it Thread 1). Thread 1 creates another thread (Thread 2); Thread 2 creates Thread 3; and so on, up to Thread 50. Each thread should print "Hello from Thread! <Number of the thread>", but you should structure your program such that the threads print their greetings in reverse order.