Seneca College

Applied Arts & Technology

SCHOOL OF COMPUTER STUDIES

Submission date: **April 17, 2023**

Workshop 10

Description:

The following workshop lets you practice basic java coding techniques, creating classes, methods, using arrays, Java I/O, inheritance, polymorphism, Exceptional Handling, JavaFx (GUI), Lambda expressions, Functional Interface, Collection Framework, Java Threads.

Task 1:

In this task you will be experiencing how to perform matrix multiplication.

- Suppose you have multiple processors, so you can speed up the matrix multiplication.
- You must implement the following method in parallel.

public static double[][] parallelMultiplyMatrix(**double**[][] a, **double**[][] b)

- Second must implemented method should be a normal sequential process. (normal method without any thread called in main method)

public static double[][] sequentialMultiplyMatrix(**double**[][] c, **double**[][] d)

- Write a test program that measures the execution time for multiplying two 2,000 * 2,000 matrices using the parallel method (by running the multiple threads).
- Also calculate the execution time for adding two 2,000 * 2,000 matrices using the sequential method (calling the method as normal call).

Hint: For parallel addition divide your matrix in to 4 matrices for example the size of 2000 * 2000 can be divided into [0][500], [501] [1000],[1001][1500],[1501][2000] then run the addition of all in 4 different threads and at the end join the threads to finish the process one after another and calculate the time of all the threads.

You can use the examples from the slides on how to calculate time for threads (week 9 lecture slides).

Note: Students are supposed to design their own output for the task

Continue to the next page…

Task 2: (Bit of challenge)

Problem Description:

There are five philosophers sitting at a round table. There is one chopstick between the two people. There is noodles in the center of the table. The philosopher thinks about the problem. When he is hungry, he picks up two chopsticks and eats two chopsticks. He must get two chopsticks to eat. The above problem will lead to a deadlock situation. When five philosophers pick up the chopsticks on their right-hand side and prepare to take the chopsticks on the left-hand side, they will be deadlocked.

There can be multiple possible solution to this situation, two of them are presented

Solution 1:
Add a waiter. Only when the waiter agrees can you take chopsticks. The waiter is responsible for avoiding deadlocks.

Solution 2:
Each philosopher must determine that the chopsticks of his left and right hands are available, can pick up two chopsticks at the same time to eat, and put down two chopsticks at the same time after eating.

Your task is to choose one of the solutions and program it is using the threads to avoid the dead lock.

Note: Students are supposed to design their own output for the task

Workshop Header

```
/**********************************************
Workshop #
Course:<subject type> - Semester
Last Name:<student last name>
First Name:<student first name>
ID:<student ID>
Section:<section name>
This assignment represents my own work in accordance with Seneca Academic Policy.
Signature
Date:<submission date>
**********************************************/
```

Code Submission Criteria:

Please note that you should have:

- Appropriate indentation.
- Proper file structure
- Follow java naming convention
- Document all the classes properly using JavaDoc
- JavaDoc should be generated properly in the project
- Do Not have any debug/ useless code and/ or files in the assignment

Deliverables and Important Notes:

All these deliverables are supposed to be uploaded on the blackboard once done.

- You are supposed to create **video with voice** of your running solution. **(50%)**
  - Screen Video captured file should state your last name and id, like Ali_123456.mp4 (or whatever the extension of the file is)
- A text file which will reflect on learning of your concepts in this workshop. **(20%)**
  - Should state your Full name and Id on the top of the file and save the file with your last name and id, like Ali_123456.txt
- JavaDocs must be used for proper documentation of each task. **(15%)**
- Submission of working code. **(15%)**
  - Make sure your follow the “**Code Submission Criteria”** mentioned above.
  - You should zip your whole working project to a file named after your Last Name followed by the first 3 digits of your student ID. For example, **Ali123.**zip.

- Your marks will be deducted according to what is missing from the above-mentioned submission details.
- Late submissions would result in additional 10% penalties for each day or part of it.

Remember that you are encouraged to talk to each other, to the instructor, or to anyone else about any of the assignments, but the final solution may not be copied from any source.