CSCI 2251
Lab 3 – Concurrent Processing

This concurrent processing assignment has the following objectives:

1. To study and implement concurrent processing, Java multi-threading.
2. To split large problems into smaller ones of equivalent (or almost equivalent) sizes, assign each to a thread to compute the results independently.
3. To integrate (gather) the results from all threads in a shared storage (memory).
4. To implement command line arguments.
5. To minimize the system resource usage, use shared memory to eliminate memory copy, multi-threading to effectively utilize processor cycles (especially for multi-cpu or multi-core).
6. To generate UML class diagram for concurrent processing and the control flow.

Description

Given two integer matrices A and B, you are requested to compose a program to perform matrix addition (A + B). Both matrices have N rows and M columns; N > 1, M > 1; You need to divide both (A and B) into four equal (or close to equal) size of submatrices ($A_{0,0}$, $A_{0,1}$, $A_{1,0}$, $A_{1,1}$ and $B_{0,0}$, $B_{0,1}$, $B_{1.0}$, $B_{1.1}$) and each submatrix has dimension close to (N/2) x (M/2). You need to create four Java threads. Each thread performs a subset of addition on one pair of the submatrices. For example, thread 0 performs addition on $A_{0,0}$ and $B_{0,0}$, thread 1 performs addition on $A_{0,1}$ and $B_{0,1}$, . . . etc. The final result should store in the matrix C of size N by M.

Note: you need to manage the case of odd number of rows and/or odd number of column matrices. Try to divide them into "almost" equivalent submatrices.

Requirements

Opens a text file, using the file name from the command line (file format as described below), and input the two matrices A and B. Divide both A and B into four submatrices and generate four threads (as described above) and assign each thread to compute the sum of one pair of the submatrices. Upon completion of all threads, gather the results and output final answer (a matrix) to the console.

File Format

1) the first line has two numbers, N and M (N rows, M columns), the size of both matrices A and B
2) the next N lines each has M elements for one of the rows of A
3) the next N lines each has M elements for one of the rows of B

Example:

```
4 6
2 3 1 2 5 1
3 1 2 2 2 4
1 2 3 2 7 2
3 6 1 5 1 3
6 5 4 1 4 3
3 3 2 2 1 1
7 5 4 3 2 5
2 1 8 4 8 4
```

For the above example, 4 is the number of rows, 6 is the number of columns. The first matrix values are highlighted in green and the second matrix is highlighted in red. The result of the sum should be as follow:

```
8 8 5 3 9 4
6 4 4 4 3 5
8 7 7 5 9 7
5 7 9 9 9 7
```

Instructions

- You must use NxM two-dimensional arrays for your implementation
- You must divide the two-dimensional array into the form such as: $A = \begin{bmatrix} A_{0,0} & A_{0,1} \\ A_{1,0} & A_{1,1} \end{bmatrix}$. Same for *B* and *C*.
- After the computation, the result needs to be stored in the matrix C.
- Your program should work for any size matrix. One of the goals is to minimize the resource usage, such as memory and processor cycles. **Explain how your code accomplishes this goal in your document.**
- All matrices, *A*, *B*, and *C* need to be in a shared memory to eliminate memory copy. They will be if you perform the multi-threading correctly!

Note:

- Work on your own
- Put comments at the top with short description of all your programs, your name, e-mail, date and course title.
- The name of the source code file must be the same as the class names. You need to have a `MatrixThread` class (file name `MatrixThread.java`) as the driver that contains the main() method, based on your design this driver may invokes other methods and/or instantiates other objects (such as the ThreadOperation object). You need to have `ThreadOperation` class (file name `ThreadOperation.java`) for your multi-threading operations.
- We expect programming assignments to be implemented using Java 1.8. Your code will be tested on the machines, Windows or Linux, installed the same Java 1.8 version (Java compiler and Java Virtual Machine). Make sure your code runs on at least one of those machines.
- Two part submissions (Blackboard):
  - Part I: MatrixThread.doc – the file contains a description on how to solve this problem, the UML class diagrams (Word document, Open Office, or others) showing the class hierarchy, control flow, and related charts/illustrations (if you have any).
  - Part II: java source code file(s) – Create a Matrix.jar file for all your java source (the .java) files, submit the Matrix.jar only.
    Example of create Matrix.jar (assume all java files are in the same folder/directory):
    `jar cvf Matrix.jar *.java`
- No late assignments. Refer to the syllabus for the late policy.

I will test your program as following example after download your MatrixThread.java

```
jar xvf Matrix.jar
javac *.java
java MatrixThread <matrix file>
```

The <matrix file> can be any file name as long as the content of the file follows the format of the example above

Example of a matrix file (matrix.txt) is attached in the Blackboard as part of this program statement. This file, `matrix.txt`, stores both matrix A and B as described in the File Format above.

You need output the result one row per line and line up the column elements (assume all elements are limited to two digit integers).