CSCI 2251 – Programming Assignment
Concurrent Processing Over a Network – Part 1 & 2

This lab is an extension of the previous lab. This lab has four objectives:

1. Integrate the previously developed concurrent processing into a client/server-based system.
2. Implement sockets for data-stream communication.
3. Establish and develop a communication protocol (what data will be transmitted in what order and how will termination of the connection be negotiated?).
4. Apply a simple graphical event-driven user interface to the client/server-based system.

Problem Description

Implement a Graphical User Interface (GUI) client-server system that performs the same concurrent processing calculations as in the previous lab.

You can think of this like a cloud computing system. The client reads in the data (two matrices to be summed) from file and transmits the data to the server. The server then concurrently sums the four quarters of the two matrices and returns the single summed solution matrix back to the client. Lastly the client displays the solution.

There will be **no** concurrent processing on the client side.

You may implement a GUI on the server side, but that is optional. You **must** use a GUI on the client side. Refer to the video showing correct interaction for more about this.

The client is also in charge of providing a textfield or textarea for the user to input a filename. Then the client reads in the data from file and transmits the two matrices to the server. The server will receive the two 2D arrays and create four threads to add up the quadrants of the two matrices. The server will then send the results back to the client for the client to display on the GUI's textarea.

The server should wait for a TERMINATE String from the client before closing the connection. That means that the server will likely need to use `instanceof` in order to distinguish between String and int[][] type data.

Note: the server should run forever, waiting for clients to request connections.

List of classes that you will write:

- `ServerStart` – contains the main method to instantiate and start up the Server.
- `Server` – the Server class code
- `ClientStart` - contains the main method to instantiate and start up the Client.
- `Client` – the Client class code, including the GUI
- `ThreadOperation` – You won't write this one because you should already have written it for the previous lab. If the previous lab went well then you will be able to use this without any modification.

Instructions for Part 1 (in three steps)

For part 1 you need to setup the GUI and write most of the client-side code.

A. Write the GUI with a textfield and implement the code so that the user can type into the textfield and see the text they typed in printed on the command prompt.

B. The user is supposed to be typing in a file name such as `matrix1.txt`. Now modify your code so that the text the user typed in is used as the filename that is opened. Open the file and read in the data (number of rows, number of columns, and the two integer matrices). This code can be copy pasted (with a few small modifications) from the previous lab.
C. Pass the two matrices to the Server and write the server so that it displays the matrices on its end (either through a Server GUI or on the Server-side command prompt).

DO NOT perform any matrix calculations on the Server side yet. Don't create threads or sum quadrants or calculate indexes. I want to make sure **Part 1** is completed successfully before you move on.

Instructions for Part 2

Integrate the concurrent programming code from the previous lab into the server code, perform the concurrent sum of the matrix quadrants, then transmit the solution matrix back to the client. The client should then display the solution in a textarea.

Make sure that you also implement the TERMINATE command to close the connection. Make sure that the Server continues running, waiting for future clients to connect.

Compilation and Execution

I will test your program as follows:

```
javac *.java
```

for the Server

```
java ServerStart
```

for the Client

```
java ClientStart
```