Project 2

The purpose of this project is to develop a client/server version of Project 1, a simulation of Remote Procedure Call communication (RPC).

The server will be multithreaded. The main server's thread takes care of the connection requests (establish rendezvous). The spawned client-helper threads will carry out the two-way communication with the clients. Most of the (already) implemented code will be on the server site.

On the client side, you will create the different types of client threads (students, teachers, principal, nurse) that will execute concurrently. These clients will ask the main server's thread to establish a connection.

When the connection is accepted by the server, the main server thread will create another "client helper" thread that will carry out the two-way communication with the client thread. The client will ask the corresponding "client helper" thread to execute sequentially the methods that were implemented in Project 1 as part of the run method. Before each method can be executed, it will send the server a message containing its name and the method name/number to be executed. This can be implemented in different ways. One way (but not the only way) would be to use a switch-case structure. This is similar to the process of creating stubs in the client and server sites.

Note that each run method should contain at least 3 methods. Some of you did not have a good grasp of Object-Oriented Programming (OOP) design concept and had only one or two significant methods in the run method. You will have to break down each thread's run method into at least 3 methods. If your Project 1 hangs and you cannot fix it, comment out the code that creates the problem. **You will not be penalized again for any mistake or misinterpretation you had in project1. If needed, clean your project1, restructure it and take it from there. The last option would be to have for each thread 3 methods with some message inside (this can be an option also for who didn't submit project1 – you can still earn up to 80 points).**

I am posting a partial example of a previous project based on a different story. It is not the only way it can be done. If it does not make sense to you, write your own way of doing it (as long as it follows the project description).

When it comes to the communication, have some message also on the client side given information about the connection request and the requests themselves (what method should be executed….)

On the server site, besides the statements displayed as result of project 1 (or simulation of project1) have message that will show when the server accepted the request and with whom.

~~1. Test your solutions by deploying the client site code in one bird and the server site code in a different bird (the most ideal situation). Ports 3000 and 5000 should be open. Check the "accounts" link for the login format.~~ We cannot do this anymore so disregard it.

~~2. If, when you test your program, you can remotely connect only on one bird, then use two windows on that bird and have the clients running on one window and the server site on a separate window.~~ We cannot do this anymore so disregard it too.

~~3.~~ The very last possibility is to run client and server on two separate windows on the local machine.

Submission requirements are similar to project 1. However, you MUST also submit a **readMe** file that will clearly mention how the programs should be tested: how I am supposed to run the client site and how I am supposed to run the server site. Keep the hardcoding to the minimum.

An additional file that you need to submit is a sample of the **output** that your program creates. As already mentioned, You should have some output on the client side as well while most of the output (following the story) will be on the server side.