Project 2
Due: Friday 27, 3PM (not later)

This project must be done individually. There will be no exceptions. 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 server-threads will carry out the extended rendezvous 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 (commuter, train, attended) 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 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 knowledge and had only one significant method 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.

I am attaching 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).

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)**

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.

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

I am usually very strict on how these programs should be tested and I allow only case 1.

However, lately we encountered so many problems with the network that I am suggesting you to start with the first case and if not working the second case and as a last alternative case 3.

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.

In conclusion they are 3 items to submit. **Each of them must contain your Lastname**

For example: Fluture_CS715_P2.zip

Fluture_readMe.txt (**make sure that following your instructions I will be able to run your project; there is no time for rechecks so if I cannot run it the first time I will not able to grade it and give any credit)**

Fluture_output.txt

Good luck !!!!!!!!!!!!!