Distribution:

Xiao: Build Server - Test Client

Linh: Build Client - Test Server

How to approach the project

This project utilizes Sockets for communication between processes. The server and client sockets are made to demonstrate a message posting system. The server maintains messages posted by clients and helps clients to communicate between each other and the client can retrieve and view messages from the server. To approach the project, we made several steps.

1. List down all messages in communication between server and client.

We discussed about the messages in communication, which format, which data should include.

2. Figure out how to store all the information, which data structure should use

We discussed about the project requirements and what data structure we should use to store messages for clients. And we created a User class to store the messages of clients.

3. Simulate a simple server where messages are enter my user, build a client.

In this step, we set up the basic framework for the project based on the example given so that the server accept clients and get connection to clients and communicate between each other. The server will run on a port which is got from the command line. Then the server will listen on that port and wait for connection from client. Client will get the hostname and port the server is running on from the command line. When a client makes a connection to the server, the server will use a while loop to create a ClientWorker thread to communicate with the client and the server will continue waiting for a new client to get connection. The client will send his name to the server and server will send some message back to the client.

4. Build complete server and test with many clients.

Added the basic menu functions to both server and client. Include mutual exclusion to handle various clients. Also set the client number limitation and message limitation for each user. Also, when invalid information was input, notice will pop out.

5. Optimization

Rewrite some methods and optimize messages.

Difficulties/interesting things

1. Storage of messages

One of the key things to design this project is that using what data structure to store the messages of users. We not only need to store the names of the clients, the connection status of the clients, the messages sent to the clients at what time from whom, but also need to make sure that when the client retrieves the data, it can be properly combined. What we did is that we created a User class and each user has several variables: name (to represent the client’s name), status (to represent whether the client is known or not), connected (to represent whether the client is connected or not), messageList (which is an ArrayList to store the messages sent to the client). And the user is stored as an ArrayList too -- userList. Every time when the client wants to retrieve messages, the userList will be checked and send the message back to server.

1. Communication between client and server

The communication between client and server is trivial but crucial. It’s very easy to make mistakes. When bugs like this case happen, it’s very hard to debug.

1. Clean code

When there’s a lot of similar function but changes on small details, it’s hard to write simple but powerful codes.

1. Debugging

Debugging is very hard, every time after changing the code, the function works well earlier might show other errors.

1. Working collaboratively is not easy, and fix bugs of other people's code is not pleasant too.
2. Deadlock many time
3. The problems have many cases and they are not easy to test.

What was learned?

1. Working collaboratively with partners

2. Using github to work.

3. Using synchronized method in Java

4. Manual testing is tedious, easy to miss cases and time wasting.

Results

Although we encountered a lot of troubles and bugs, finally we found them and fixed them. The output of our code matches the output of the example given.