EE 422C Project 7
Chat Room with Network Programming
Pair Programming

See Canvas for due dates and further instructions. This is an optional pair assignment.

General Assignment Requirements

The purpose of this assignment is to use Socket Programming to build a simple chat room application. You are free to use whatever classes and methods from the JAVA 8 library you wish. You may not use non-standard library features.

In this assignment, you are asked to implement a simple chat room application with both Server and Client. You should implement necessary UI in JavaFX, and use JAVA Socket for network communication. Design your application using O-O principles. For full credit, you must also use the Observer design pattern using the Java Observable and Observer class and interface.

Server: The server is responsible for receiving messages from clients and dispatching messages to appropriate clients. You just need one server; call the main class of the server ServerMain.java. Make sure that ServerMain.java has a main() method.

Client: You should support both the chatting scenarios below. They should work with at least three clients, and (for full credit) two different computers.
- ***One-to-one chat:*** Client A can chat with Client B and Client C individually at the same time. You can just use one window to show both chats. Using separate windows to represent different conversations is also acceptable.
- ***Chatting Group:*** Client A can create a chat group, and send message to B and C in the same chat group, and all members in the chat group can send/receive messages to/from the group.

Make sure that you have a ClientMain.java file with a main() method in it.

You are encouraged to design the UI based on your experience of using chat room applications. You are encouraged to add more interesting functions to the chat room, such as registering with a login and password, retrieving chat history, sending friend requests, only allowing friends to chat with each other, and providing a UI to change password. These extra features can be worth up to 25 extra points based on their complexity. Please discuss your ideas with the TAs before implementation.

For full credit, bundle your Server code and your Client code into executable jar files, that we can execute by double-clicking. There should be one jar file to start up the server, and one to start up the client.

Extra Credit for failing students

For most students, the grade will be capped at the point value on the assignment on Canvas, although all of you are encouraged to make your program fancy. However, for those of you who are failing the class (grade of D+ or lower), the extra features you add will serve as an extra credit project, and may be used to push your grade up to a pass. To earn these points fully, use good O-O techniques and good data structures, implement a good GUI, and have a lot of useful features.

Submission and grading

- UML Diagram: Use case Diagram, and Class Diagram.
- README.txt file with
  - Your names and UT EID's,
  - a list of files that you are submitting,
  - a description of your high-level testing, including whether you tested on Windows, Linux, or MAC,
  - Git URL
- Put your code in a package named assignment7, zip all your files and name the zip file as Project7_EID1_EID2.zip (see below). Do not turn in your test code. Have a header for every file.
- Include your two executable jar files.
- Check in your files regularly into Git. We expect at least 3 substantial check-ins from each team member. Individual students are also encouraged to use Git.

- Each team should also provide a document `team_plan.pdf` describing the work done by each of you. This document must include your Git repository URL, and your full names and EIDs.

Make sure that the structure of the final ZIP file is as follows, when unzipped (same as Project 4).

```
Project7_EID1_EID2/ (folder that is created by zip)
    README.pdf
    team_plan.pdf
    <other non-code files>
    <executable jar files for server and client>
    src/
        assignment7/
            ServerMain.java
            <other code files>
```

We will ask you to demo your implementation with at least 3 clients and 2 computers during the TAs' office hours.

Before submission checklist:

- ☐ Did you complete a header for *all* your code files, with both your names and UT EID's?
- ☐ Did you mark the slip days used?
- ☐ Did you do all the work by yourself or with your partner?
- ☐ Did you zip all your new or changed files into a zip file? Did you remember not to include your testing code files?
- ☐ Did you download your zipped file into a fresh folder, and run it again to make sure everything is working?
- ☐ Did you complete the README.pdf file with all the requested info?
- ☐ Does your code work correctly on Eclipse with Java 8?
- ☐ Is your package statement correct in all the code files?
- ☐ Did you preserve the directory structure?