Chat Application

Project Report

A basic chat application built in Java using Eclipse has been completed. After implementation, the design started to change from what the document I had created was outlining, which makes it obvious that keeping documentation up to date during implementation and testing is difficult. My application has been split into two main components and an auxiliary module for the user information all coded in Java. Because this is a client-server model type of project, the focus was on these two objects.

My server object has been hosted on the localhost on a hardcoded port. From this point, the server will then start creating server threads and its main job is to manage the communication between these threads. Once a client has connected, a new thread can be created and manage that client. The design made sense to separate each client to have a thread to manage its actions and simplifying the communication. For my project, I have chosen to use a buffered stream of strings to handle the transactions between the client and server. Each different action will have a unique string and the client or server will handle it accordingly.

My client object will be started after a user has successfully connected to the server and logged in. The GUI for the project only provides the client with a view (not the server which only has some standard console output for debugging purposes). To assist sending of messages, an interface of MessageListener was created so that each client could add these listeners upon a status event(login/logout) and that will also help populate the active users list. The user list was assisted with another interface of the UserListListener. From here the GUI element for which users are logged in can be modified by the client and changed according to status events.

Because I had taken the communication being a buffer approach instead of having an object being passed between the server and clients I ran into some interesting hiccups. At first I had chosen to create the GUI and it was started with private (one-to-one) communication. When the group chat aspect was added, a new method was needed to be added to the MessageListener interface to differentiate between a one-to-one or a one-to-many message. Any changes to these protocols ended up changing an entire aspect of the program unfortunately. For example, when I was attempting to change how the one-to-one messages were being sent to users, the one-to-many messages completely stopped working and had to be debugged. Other challenges encountered include working with swing gui and realizing that the resolution difference of computers makes maintainability a big concern if this project was to be extended to become a networked application.

Test plans for this are based upon basic functionality execution based testing. Input for all of the modules has expected positive and negative tests. The test results have been added to the github repository with their results.

To run my program:

I used a third party library from Apache (Apache Commons Lang 3.5 https://commons.apache.org/proper/commons-lang/download_lang.cgi) for string parsing. This

will need to be included into the project directory through a build path (I used eclipse). The JAR file for the library is included in my github repository for the project.

Load the .java files from the github, including the two .txt files (users.txt and pass.txt). Without these two files the user store will not work. Other text files are not necessary (for the history yet, because they can be created by the program). The .txt files must be stored in the local project directory.

Once the project files are moved into the same project and the apache library is included, then to run follow this sequence->

Run the “ServerStart” class.
Once this is up and running, run the “LoginFrame” class.

The LoginFrame class will start new clients and can be used as many times as needed. Any other files running will disrupt the program so only start these two (LoginFrame can be run more than once and ServerStart only once at the beginning).