Summary

General Approach:

Since this was a team project, the initial approach happened in two different ways. We first decided to use Github to collaborate between each other and have our project in one central repository that we could share. After setting that up, we began coding what we knew that we could do easily.

James’ Approach:

I had been working in C almost 2 years without touching Java, so this required time on my part of searching the APIs on the Internet and looking at the example codes. Nic also helped me understand how to use threads since he did the previous project. We both planned out what parts of the project we would do and agreed that we would work on separate parts of the menu system, implementing the Client and Server side functionality as we do so. After Nic had gotten the threads up and running and PrintWriter and BufferReader. I was able to start cranking out menu functionalities. Java made things extremely simple. I also helped Nic identify problems along the way and we both worked on fixing bugs together aside from our separate work.

Nic’s Approach:

I First made the starting java programs, and the make file so that it would run correctly. Then, I made a very simple menu that would be communicated using the PrintWriter, BufferedReader, and other java constructs. After that, I turned the server java file into a runnable and then used inner classes that I learned from the last project. I was then able to get the multiple server threads running and was able to set up a basic switch case for the menu items. Then I did the menu items that I needed to do and was agreed upon and fixed up some problems

James’ Difficulties:

The project was not terrible difficult to me. The main difficulty I had was re-learning the API’s for Java since it had been so many years since I had used them. Once I learned how to do the API’s, the rest of the difficulty was formatting of the menu system and passing messages between the 2 different programs.

James’ Lessons Learned:

The main thing that I learned was that using InputStreamReader, PrintWriter and BufferReader is all that is needed to pass messages between programs in Java using sockets. The socket is almost self-explanatory. We could simply use the socket input stream in the InputStreamReader constructor and the socket is set up for an input stream. Then we could use that input stream in the BufferReader constructor and have the final setup of the input. The output could easily be achieved by simply using the socket output stream in the PrintWriter constructor. After this, the entire socket communication back and forth is ready to go. Java made me realize how much it hides from the users.

Nic’s Difficulties:

I had no idea what to do to coordinate with the messages, so I made the decision to do something instead of nothing. I made an arraylist for the users and an arraylist arraylist for the messages, which was a little too hardcoded for my liking but was at least something that worked. James later told me to use a hash map instead. I had a hard time with keeping multiple users logging in with the same name and the various annoying things that the users could do. I also had a hard time figuring out how much to put in the mutex and then realized a few things that could get messed up with concurrency of writing, so I put the semaphore around that too.

Nic’s Lessons Learned:

I learned better what to use semaphores around, and that any code is better than no code, as long as it is in the same sprint or iteration and hasn’t been submitted permanently yet. I also learned how servers and clients work and that it is important to send as little information across as possible. I learned many structures that are used for certain things.

End Result:

The end result was something that we were both satisfied with. We managed to work out some major kinks in the program, such as having a user show multiple times in the connected users list as well as not deleting them from the list properly when the user chooses to log off. We also realized that we were not handling cases for null inputs in the menu choices and allowing null names to be entered for users who were connecting. We fixed these cases but we decided to allow the user to enter numeric and alphanumeric names since a user on a computer can be successfully identified by those. This thought came from the idea of email addresses, which do not necessarily contain the user’s real name. In the end, we chose to keep the ArrayList of Arraylists of messages instead of employing a HashMap. This was because of the implementation was already focused on the ArrayList. It was a compromise we were both willing to accept since this project is academic and does not care about efficiency. All functionality was completed in the end and the result was satisfactory.