Final Report: CS 300 Messaging Application

Jared Weakly

June 8, 2017

Code Structure and Program Challenges

The code is structured in a fairly simplistic way. I tried to go as overly simple as possible without ruining the integrity of the application's functionality; I think this approach served me well. For starters, there is a Client class, a handler class, a "protocol" class, a Server class, and a User class. The User class is there so that I can contain and organize all of the information about a certain user throughout the lifetime of the data; the Handling class got split out into its own file because it was getting a bit cumbersome to keep inside the Client class. I implemented a simple-ish parser and it was, quite honestly, a lot more painful than I expected it to be. Most of the pain I encountered was from trying to deal with concurrency/threaded application behavior.

Talk about any special features developed in the application above and beyond the requirements.

One thing I'm particularly proud of is the fact that I implemented all of my client/server communication using a dummy POST api style. This style uses an enum to dictate what kind of data the client is about to receive or send and how to process it accordingly. Because of this, I was able to achieve some very high degrees of type safety for this sort of thing (at the cost of having to fight the compiler sometimes); in the end, I was rewarded with really stable code. I noticed I tended to have much less issues with making sure things went in the right place because of how I setup my typesystem. I also implemented single user messaging through a ".query" functionality, very similar to IRC.

How to run program.

- Step 1: Open up project in Intellij.
- Step 2: Execute Server.Java
- Step 3+: Execute Client.java, follow the pop-up prompts, and use clients as normal.