David Fantin

ID: 1525813

CS-4333

9/29/2020

**Networks Project 1 Report**

**(Oversimplifying) The Design Process:**

Throughout this project I went though many iterations hon how to merge the client and server code. My first idea was to just have two separate sub-classes, one for client, and one for server. I was able to get this to work, but only one way (Client → Server). I then decided to combine them further and make common variables for each, then declare the necessary ones in each sub-class as needed. To do this I made another sub-class that would call the individual client and server classes. Before I settled on this structure I did go on a tangent combining all three into one class and using additional if-statements instead, but that became too complicated. I resorted to the three sub-class method. Then I decided to deal with all the possible commands that can be inputted. This was my first time writing a program to accept multiple command line inputs, so it might be a bit clunky. I made three separate if-statements in the parent classes main method which checked the “args” value. Then depending on the output I started the parent class of the client and server class by calling “TClntSrvr.TCS(ARGS);” (after setting ARGS = args, so I could use it though out the code). And set int p = to the location in args that the port number is located (for future reference when starting the connection). To make the Auto Mode work, I set up an if statement in the catch IOException of the client class to increment a variable if -a is the first argument which allows the final if statement in the TCS method of the TClntSrvr class to launch the server class if a has already gone through the client. At this point I had everything working except that the server could not send messages to the client. So I tried copying the code that allowed the client to write messages to the server class and the code that accepted messages to the client class. This worked… however it was clunky because of the readLine() methods. I tried to implement the isready() method from the video, but I could not get it to work, so instead I decided to make two thread classes (one for input and one for output) and have the client and server classes to just start the two threads after a connection was made. It worked!! The last thing I had to do was set the default values which was easy, I just created two new variables and had if-statements to check if the port or host-name was given, if not, use the default values. For the port number I used 12987 as instructed and for the host-name I used 127.0.0.1 which I believe is the “localhost.” In addition I included code which is commented out that allows the default host-name to be the client’s actual IP address.

**Feedback:**

I really liked this project, I feel confident about coding with Sockets in Java now, as well as how simple bidirectional communication works.