Derek Huang
20022672
February 28, 2021

CISC 335 Programming Assignment 1 – Report

a. Brief description of the code steps, operation, and considerations (if there are any considerations).

This piece of code is written in Java. There are 3 classes in total in my submission. They are respectively XServer, XClient, and singleConnectionHandler.

XServer is responsible for taking charge of the server side. The server can accommodate at most 3 clients. So, I created a fixed thread pool size of 3 to fit this requirement. I then initiate a server socket on port 8563. I use a forever for loop so that as long as the loop holds, we accept client's connection. Until the number of clients reaches 3, we say that the server is full and close the connection.

XClient is responsible for taking charge of the client side. The client will make a connection on the port 8563 so as to connect with the hosted server. Within the client class, we ask for input from the user via the command line input. User can either enter "exit" to terminate the connection with the server or enter "list" to list all the files stored in the "files" folder in the directory. We finally turn off all of our connections.

SingleConnectionHandler is responsible for taking charge of what a single client's work would do. It inherits from the Thread class. We have an override method run() here. It handles information such as what the client is, what the date the connection was made, as well as what the user has typed in (input from the user via the command line input). As the user type "list", it takes each of the files stored in the given path folder, and list its name one by one until we hit the end of the folder. As the user type "exit", we terminate our connections. Other reply would be acknowledged (ACK), but not executed.

There are two files are stored in *cisc335_A1\files*. One is of type .jpg, and the other is of type .txt.

b. Difficulties you faced and how you handled them (if any)

One of the difficulties that I faced was the restriction on limiting the maximum number of clients to be able to connect to be within size 3. I first couldn't figure out a way to do so, but I later found that it might be a good idea to make use of thread pool. That's the reason I created an ExecutorService fixed thread pool of size 3. Afterwards I created another class named singleConnectionHandler which extends from thread. The fixed size of the thread pool can help me limit the max size to be 3, whereas the singleConnectionHandler can help me handle a single connection of the client with the server, more systematically.

c. Possible improvements: If you had more time, what would you add or do differently?

If I had more time, I would actually try to handle more clients to see if our server's capacity can accommodate how many clients at most. From this end, I can see the throughput and availability of the server. Furthermore, if I had more time, apart from listing the files, I can try to open it. For example, as the user type open "XXX.jpg", the program will open that file immediately. Or if the file is in .txt format, we can display the user the content in the console.