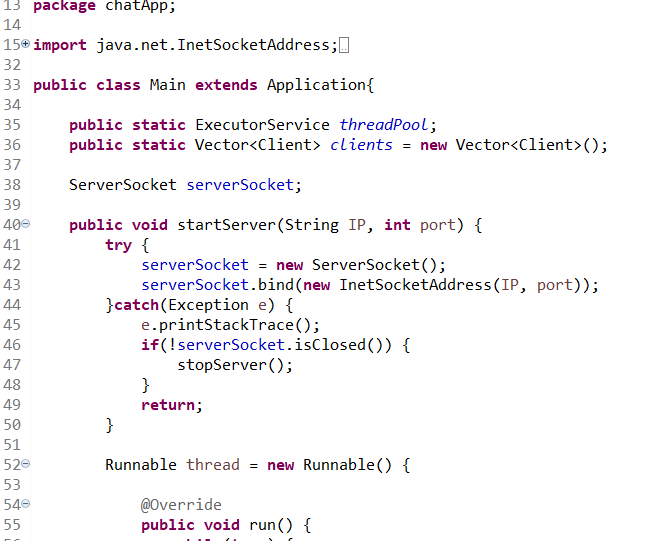
Name: Daryan Chan

ID: 113973192

Section NDD



For this main class, I made variables that I would use, such as clients, threadPool and serverSocket. Then I created a method to start the server that will take the IP address and port. Inside this method, I did a try catch method. Inside the try, it will create a new ServerSocket object and it will bind to the specified server address and port. The catch will handle the exception but will also be in charge of handling closing the server. There is also a Runnable thread here and will override the run method.



The purpose of this is so that the server will continue to run as long as it’s true. Therefore, it will always continue to go through the try catch method and connect clients to the server. Executor and newCachedThreadPool is utilized to create a new thread pool that creates new thread pool as needed. Then it submit a runnable task to ExecutorService.

The stopServer is self explanatory and that when called, it will go through the Client using iterator and close them.

The start method is basically in charge of creating the GUI interface. Since I’ve explained it in previous workshops, I won’t go in depth but the purpose of it is to allow server connection.



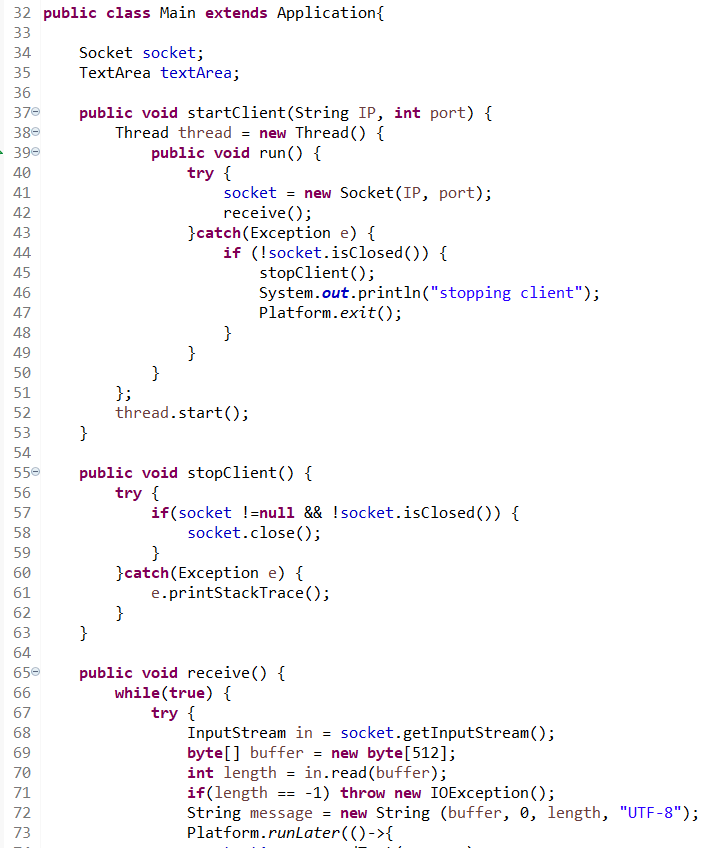
I also initialized the IP address and port number. The action events was utilized so that certain functions, such as starting the server will be initiated when pressed.



For the Client class, this is to create the client and server connection. The constructor will initialize the Socket and will call the receive() method.

The receive method will start a new runnable and another run method. This will continue to run as long as it’s true and will store the input into the byte array and will get the length of the byte array. If array length is -1, it will throw exception. It will then go through for loop for each client and send the string message to the client. The threadPool.submit method will then be called with the thread in the Main class.

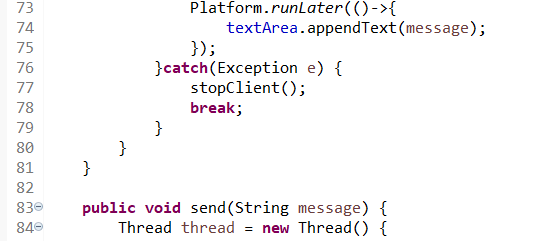
The send class is similar to the receive class but will uses the OutputStream instead.



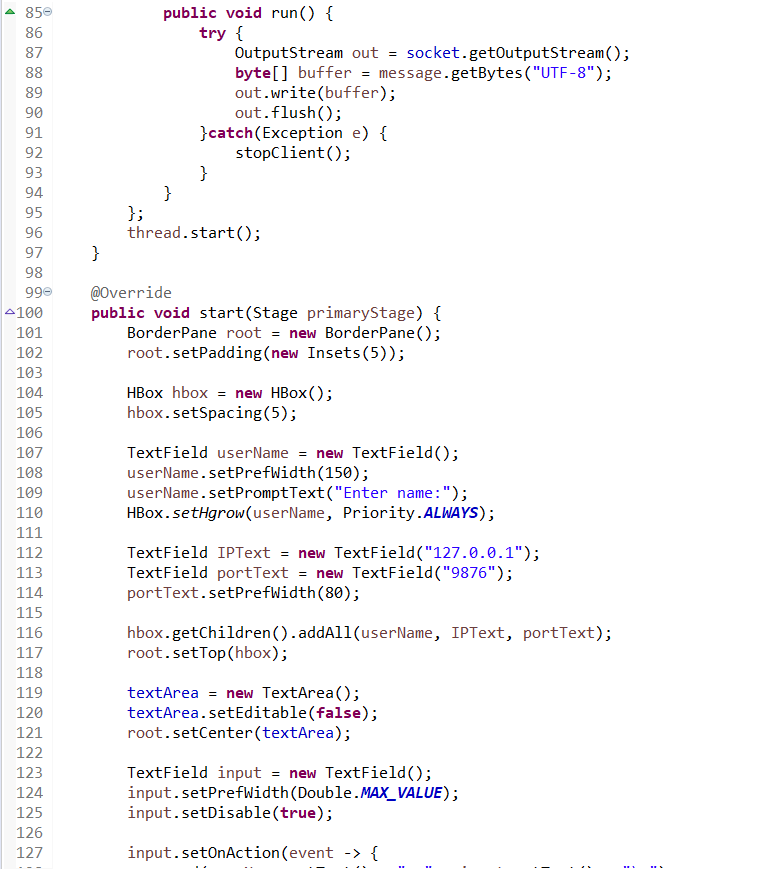
For this class, I created startClient method that accepts IP and port number. This is necessary so when client starts up, it will be connected to those fields and be able to receive data. This will be done in try and catch method too and will stop the client if exception is thrown.

The stopClient basically close the socket if socket is not null or not closed.

Receive method has the same concept as explained previously.



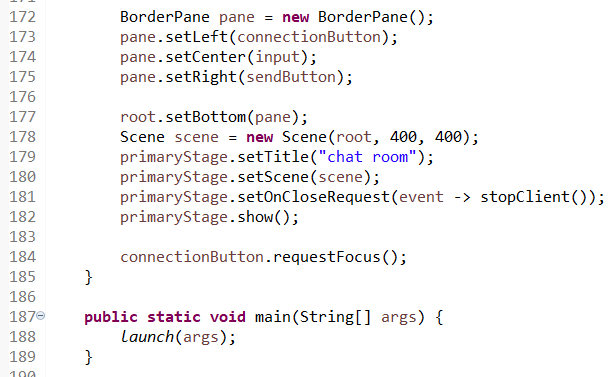
TextArea will be append to show new information based on the message. If exception is found, then it will stop the client and exit.



The start method will create a GUI using javafx for the chat app. The code is pretty self explanatory so I won’t explain in depth but it create text areas for user to see the message and textfield to see all the relative data, such as address and port number.



Button interfaces are also created so connection and sending messages can happen.



Last part of the codes basically set everything in motion and creates the GUI.

