ITCS 6166 Project 1: Socket Programming Multi-user Chat

Overview:

In this project I have developed a Multi-user Chat Room using the client-server model. This is achieved through creating multithreaded classes and sockets. Users are able to login with a name of their choosing. Then they are able to send messages to each other. Users are able to leave and re enter the chat room without any issues. Lastly the chat room text is stored in a text file, allowing the retrieval of previous conversations if the server needs to restart.

Classes/Methods/Objects/Variables used:

Client GUI related:

I will not cover all of the individual GUI variables since they are relatively self explanatory

buildLoginWindow:

This function is used to generate the login window. Users are able to assign themselves a username before they connect to the server.

buildMainWindow:

This function is used to generate the chatroom. This function also includes the configureMainWindow and mainWindowAction.

configureMainWindow:

This function is used to generate and style all of the components inside of the chatroom window.

mainWindowAction:

This function is used to assign actions to all of the buttons.

loginAction:

This is the function that runs when the enter button is clicked. This will send the user's name and run the connect function.

Connect:

This function connects the user to the server. This will also start an instance of the clientRunnable class.

ACTION_B_SEND:

This binds the SEND function to the send button.

ACTION_B_DISCONNECT:

This binds the DISCONNECT function to the disconnect button.

clientRunnable:

DISCONNECT:
This function tells the server that this client is ending thus the server needs to remove this user's name and socket from the server. The message includes #?! At the beginning to let the server know this is an action and not just a message.

checkStream:
This function is just an infinite while loop to constantly run the RECEIVE function.

RECEIVE:
This function is used to receive messages from the server. The function will check if there is any input coming from the server. If there is input coming from the server, first check if the message contains "/#!". If it has this then this is chat history. If the message contains "#!?" then this information should be used to populate the user list in the GUI. Otherwise just display the messages normally.

SEND:
This function is used to send messages to the server which will be displayed to other users.

Server:
ArrayList<Socket> sockets
This keeps a list of all the sockets that are currently connected to the server

ArrayList<String> users
This keeps a list of all the users that are currently connected to the server

addUserName:
This function is run whenever a new user enters the server. When a new user enters the chat, echo the updated user list to all users. This way all users will have the updated list.

printChatHistory:
This function is run whenever a new user enters the server. For the user that just entered, the server will read through the text document and add those messages to the new user's chat.

serverRunnable:

checkUsers:
This function constantly checks if the number of users has changed. If the number of users has changed, update everyone's user list.

writeToFile:
This function runs whenever a message is sent to the server. When a message is sent, and if it is not a command. Write it down to the chat.txt file.

sendToClients:
This function is run whenever the server receives a message. When the server receives a message, send that message to all clients.

removeSocket:
This function is ran if the message sent to the server contains #!?. This means that a user has left the chatroom and must be removed from the socket list and the users list.

Program Structure:

Client:
From the client side, the main function will begin the generation of the main window and beginning a thread of the clientRunnable class. Once the user connects to the server they will have access to the send and disconnect functions through the GUI. The GUI will constantly check if it needs to update based on what is coming in from the server.

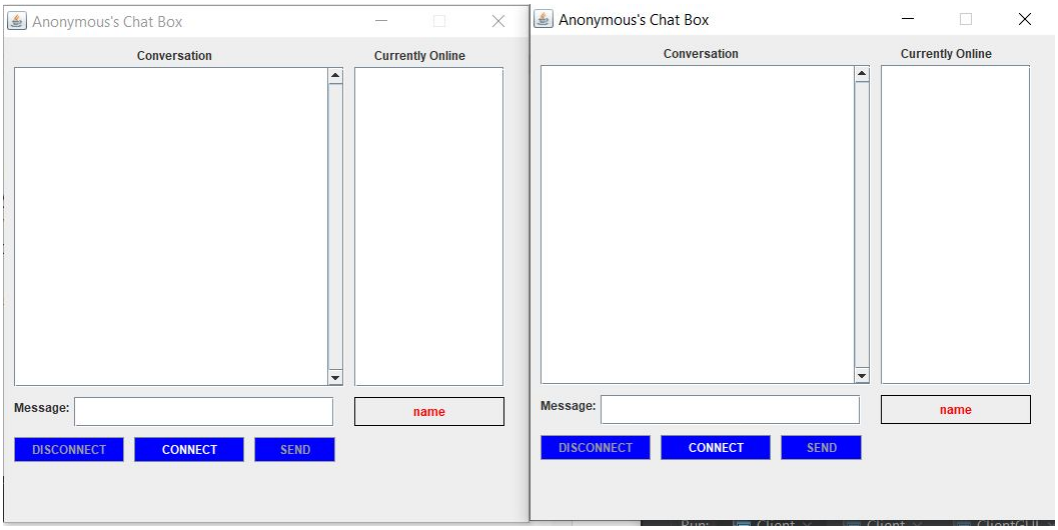
Server:
For the server side, the main function will begin the generation of the server socket. Once the server socket is generated, the server will constantly check if any clients try to connect to the server. The server will also start a serverRunnable thread. In the thread the server will check if it needs to write to a file, remove a socket, or send data to the client.

Example Case:

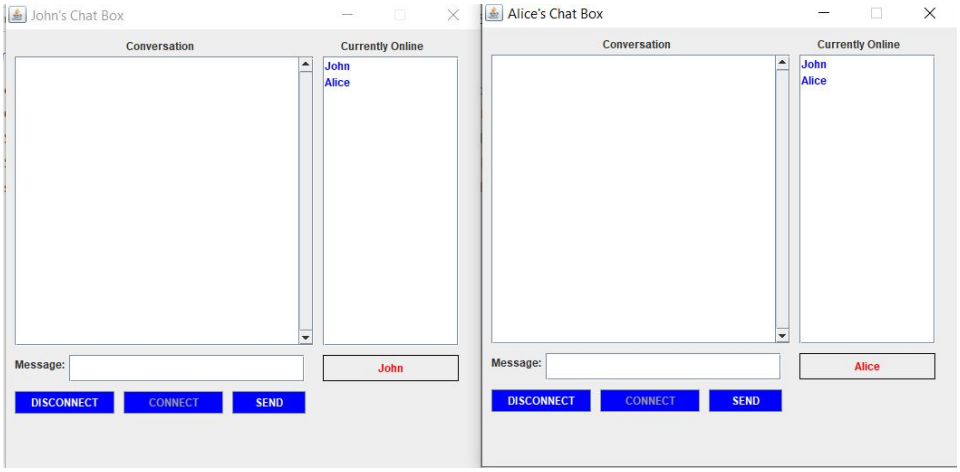
First we launch our server

```
C:\Users\sexyviper69\IntelliJIDEAProjects\ITCS 6166 Programming Project 1 HTTP Client and Server\src>java Server
Waiting for clients...
```

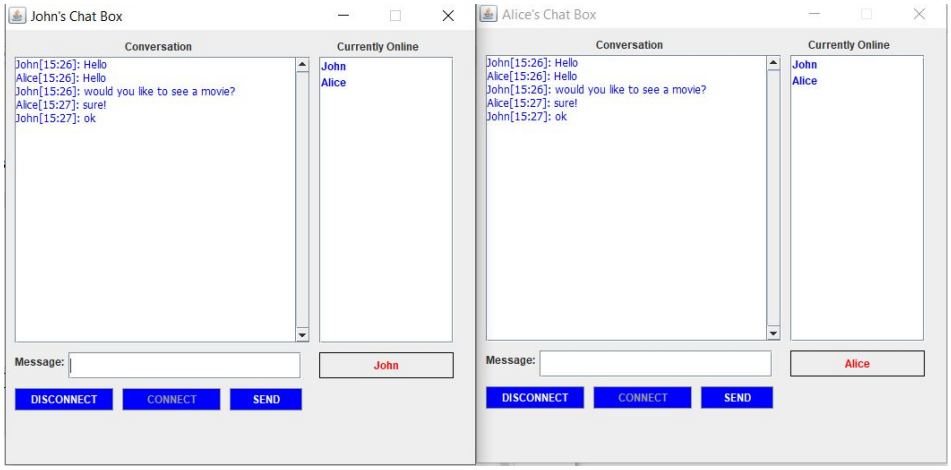
Next we create a few clients.



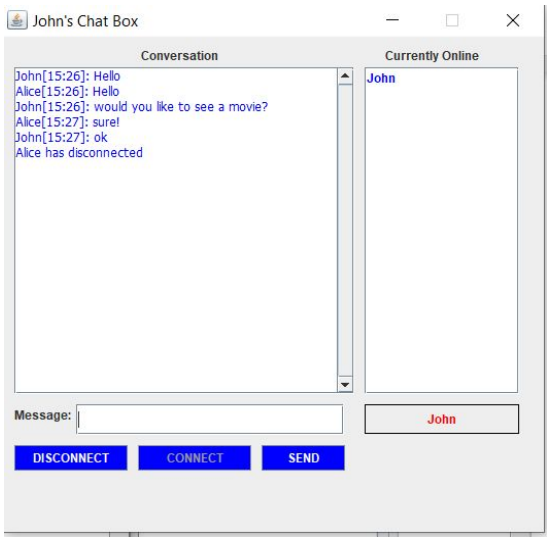
Then we connect to the server using both clients



Next send some messages



Once Alice disconnects, John's client is updated



Now John leaves and Alice reconnects. You can see that the chat log has been preserved.