Computer Science Cluster



**COMP20081: Systems Software**

**Project report:**

**“Music application” – Internetworking with Java**

**Produced by:**

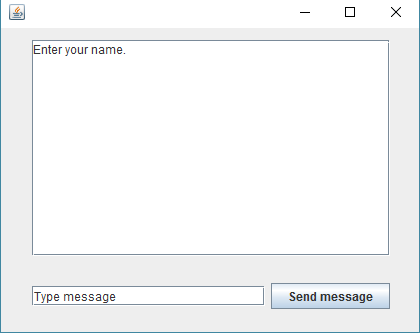
**William Crane (N0637943)**

**Hayden Palmer(N0641991)**

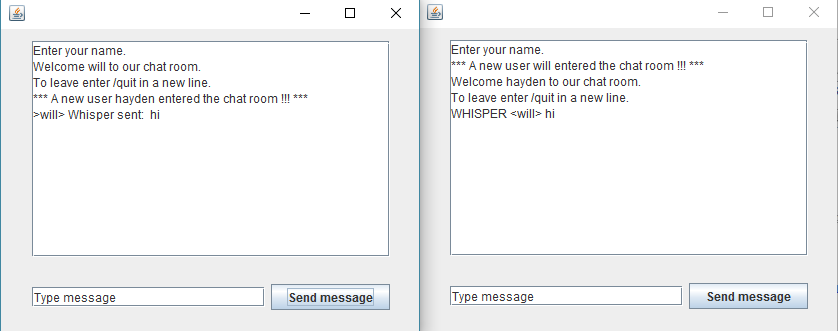
**Module Leader: Michalis Mavrovouniotis**

**Included in the client:**

Within the client we have a register feature which is requested at the time of opening the client, this will allow the user to create a custom username while they are connected to the server itself. It is requested as shown below, and held server side but not client side.

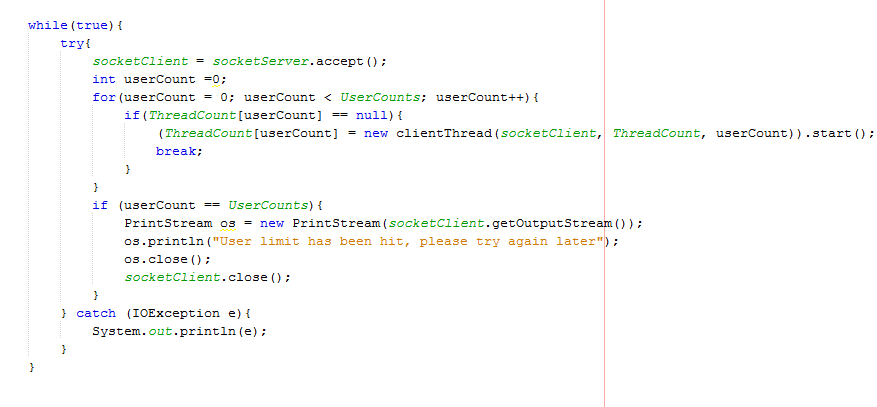


Within the client you have the ability to send messages and whispers to specific members that are currently connected to the server, while also notifying the user who receives the whisper that it is a whisper and also who it is from, as shown below:

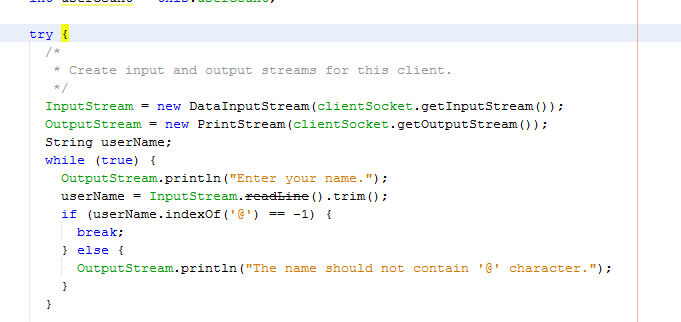


**Included within our server:**

Within our server we hold the ability to add any value of users to connect \*we decided to default to 15\*, this means that you can change the server to be able to handle more clients at a time or lower amounts depending on your needs, higher client counts will be more taxing on the system itself though.

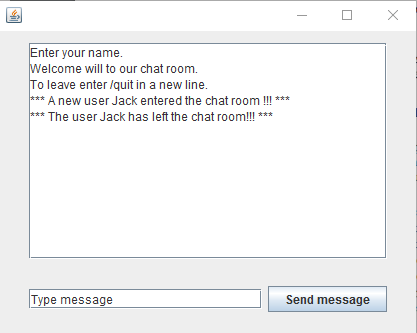


Once a user has been accepted onto the server it will handle adding them and logging them into the server, to do this it will send a request for a username and will reject any username that has an “@” symbol at the start, this is to avoid the whisper system being toggled and therefore not sending the message correctly.



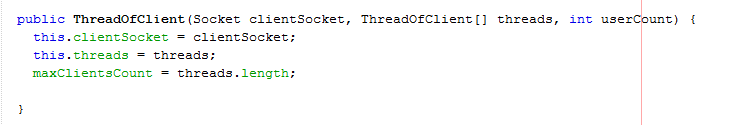
Once the user has been connected to their thread and logged in they hold their own thread on the server to allow for them to talk at their own pace with the people connected to the server be it via whisper or via public chat.

Once a user connects / disconnects it will notify the other users on the server of the change in the server of the new user connecting / leaving, which can be seen in the GUI image of the client below:



How the connection between the server and client is formed:

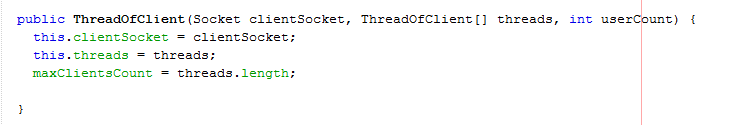
The server has a limited number of slots for the user to connect through, the user once connected gets assigned a thread ID, this will allow for them to maintain their connection / come back with the correct details assigned to their account:



The connection held to the server is connected on the fly, if the server needs to be contacted by the user or the server needs to contact the server then the details will be sent over, this means that when a message is sent on the server it will send the message over all of the threads available.

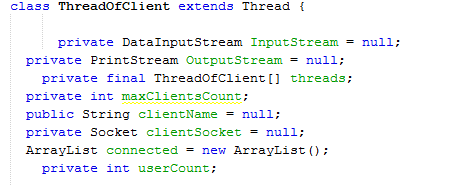
The way that the connections is handled is by checking through each of the threads available if there is a user on the line, if there is it will send the message out to them from the server once a message is received by the server itself.

How the server stores the data of the users:



Firstly it assigns a thread to the user, giving them a place to store the data that they hold, once this is done the user will give their username.

At the start before any data is traded the thread holds these details of the client connected:



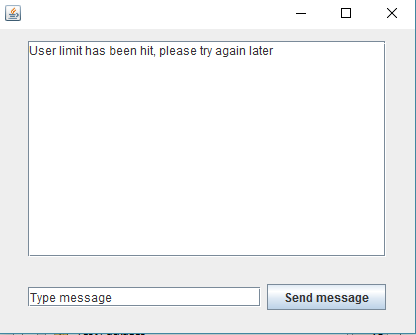
Allowing for the thread to store the details required for the client to come back and keep communicating with the server while they are logged into the server itself.

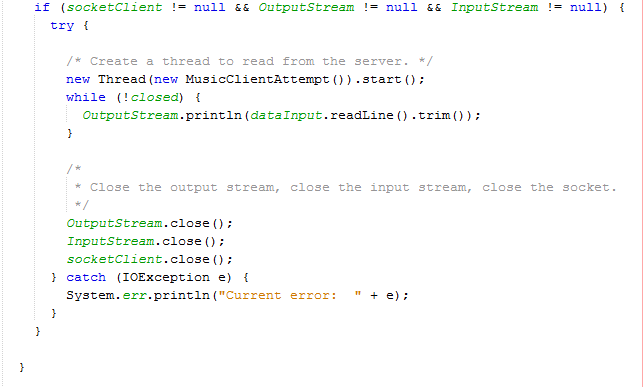
How the client contacts the server:



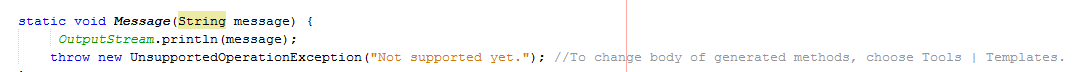
Before the user can do any interaction with the server it tries to connect to the server, if it is able to find a server it will open the GUI to start the process, if it unable to find the server that it is looking for then it will end the program and print out an error in console.

If there are too many users already connected to the server the server will kick back an error telling the user that there are too many users connected to the server at the current time while also closing the socket with the client.



The client then creates a thread for it to read from the server if it is connected while also closing the thread if it is unable to connect, this is done through this part of the code: 

While the user is connected the main function is the messaging function given the server is the side where most of the details are done, this is done by using the outputstream created at the start of the program once the connection is confirmed.



The connection to the server is broken once the user has detected a message which begins with “\*\*\* Bye” , therefore saying that the user has typed /quit for the server to close the thread down.

