Name- Shivam Kumar Sareen

Student ID- 1001751987

****

About:

****

LAB 2- Asynchronous Message Server

It is a multithreaded client/server Chat application based on a simple GUI which uses Java Socket programming.

A server listens for connection requests from clients across the network or even from the same machine.

Clients know how to connect to the server via an IP address and port number.

After connecting to the server, the client gets to choose his/her username on the chat room.

The client sends an asynchronous message, the message is sent to the server using I/O stream in java and the client doesn't wait for the acknowledgement from the server.

Java object serialization to transfer the messages.

REQUIREMENTS:

IDE used- Eclipse

1) Server:
    a) Should support at least 3 concurrent clients.
    b) Display details such as connected clients and the ongoing process.
    c) Should notify other clients about uploaded message.
    d) Allow other clients to receive the message.
    e) Logout from the server once the Logout button is pressed.
    f) Should not fail on unexpected crashes.

2) Client:
    a) Ask for username to log in. Reject if duplicate.
    b) Notify on getting connected.
    c) Sends the message using one of the messaging options (Broadcast, Multicast, Unicast)

There are 3 modes of transfer of Data(messages) within clients-

- Unicast

- Multicast
- Broadcast

Broadcast- After receiving the message from the client, the server broadcasts the message if it is not a private message(message doesn't begin with '@').

Unicast-And if it is a unicast message which is detect using '@' followed by a valid username, then send the message only to that user.

Multicast-In case of Multicast, the message is sent to all the specified users, if found. The message for this is written by including '@' followed by all the usernames separated by ','

Receiving the message

Since this is an asynchronous mode of messaging service, the client does not wait for the acknowledgment from the sender and needs to click on the Receive Message to receive the message.

Persistent Messages

The messages stored are persistent i.e. the client messages are stored and can be read even if the server restarts.

****

Instructions:

****

Server

1. Run Server_GUI.java file

2. Click on the 'Start' button and Server will start on the configured localhost and port

Client

1. Run Client_GUI.java file

2. The Client will connect to the server, if the server is server was previously started, else it'll show a message saying the.

3. Enter the Username for the client to connect to the Server

4. If the username is unique, the client will start the connection with the server, else it'll ask the user to enter a different username.

To implement multiple client chat, re-run the Client_GUI.java file

Chat

While in client console:

1. To send the message, there are 3 options-
    a.) Simply type the message to send **broadcast message** to all active clients
    b.) Type '@username<space>yourmessage' without quotes to send message to desired client as a **unicast message**.
    c.) Type '@username1,username2,username3<space>yourmessage' without quotes to send message to desired clients as a **multicast message**. The number of user specified can be n in number. In this case, n=3
2. To receive the messages, the client clicks on 'Receive Message' button.
3. Click on 'Show Connected Users' button to see list of active clients
4. Click on 'Logout' button to logoff from server

Known Shortcomings

- The server is responsible for all the operations. Thus, if the server is closed before the clients, then the client status won't be able to communicate with the user. This might lock out the clients the next time the server is up until it is manually terminated.