Project 1: A Client-Server Chat Program

Stefan van Deventer
20273401@sun.ac.za

8 August 2018

1 Introduction

The purpose of this project was to create a client-server chat program that can handle multiple clients at the same time.

2 Unimplemented Features

None of the required features were unimplemented.

3 Additional Features

Added a help button and some user experience improvements i.e. being able to press enter to send a message instead of clicking send.

4 Description of files

The program consists of 4 java files:

- ProjectServer.java
  - ProjectServer.java is to be run as the server. It accepts client sockets and validates the nicknames for each client. If the nickname is valid the server adds the clients nickname and socket to an arraylist with the other clients. Then a thread of Chatserver.java is started for the client and ProjectServer accepts the next client if there is any.
- Chat_sever.java
  - Chat_server.java is the instance of each clients server send operations. If the client sends a message this is where it is processed sothat the server can send the message to the other clients appropriately i.e. when the client whispers to another client the server doesn't send the message to every client.

- Client_GUI.java
  - Client_GUI.java handles the GUI for the client. The GUI is built and the connection to the server is made here. When the client chooses to connect via the connect button, the Connect() method is called and the client connects to the server with a specified address. Then a thread of Client.java is started for the client.
- Client.java
  - Client.java handles the receiving of messages for the client, the sending of messages, the disconnecing of the client and the population of the current online users list. A thread of each user runs the receive method sothat the client can always receive a message. The other methods are called when needed i.e. when the user clickes the appropriate button on the GUI.

5 Program Description

The program flow is as follows.

1. Start the server on the machine by typing 'java ProjectServer' in the command line. The server will then enter an infinite loop to wait for clients to connect to it.
2. Start a client on the same or any other machine in the network by typing 'java Client_GUI <host address>' in the command line. Thus the client will know where the server is that it needs to connect to. Click the 'Connect' button on the GUI and choose a valid nickname. This request is sent to the server where the nickname is validated. If the nickname is valid a thread of Chatserver.java is created for the client. If not the server sends back a message that the nickname is invalid and the client must choose a different one. The process is repeated until the client has a valid username. When the connection to the server is accepted, the ClientGUI creates a thread of Client.java sothat the client will always receive messages, even while busy with a different operation.
3. When a client desides to send a message, they type the message in the text box provided on the GUI and click send. The send button calls the send() method in Client.java where the clients name is added to the message. The message is then printed to the output stream and the server receives it. When the server receives the message it analyses the message to see what type of message it is i.e. a public message or a whisper. The server then sends the message to the appropriate clients.
4. When the client wants to disconnect, they click the disconnect button. This calls the disconnect method in Client.java and prints the disconnection message to the output stream. The server receives this message and

removes the client from the list of current users and closes the socket. Then all other clients are informed that the client has disconnected and they update their online users list. The client closes his socket and exits the GUI.

6 Experiments

1. Will the program crash if multiple users connect at once?

   - To test this I opened multiple clients and connected them to the server at the exact same time. All the clients were connected as expected and the server did not have any problems handeling it. The list of online users for each client was correct and message sending worked as intended, thus no data was corrupted.

2. Will data be corrupted if multiple clients sent messages at once?

   - To test this I sent messages from multiple clients at the same time. The server handled it as expected and the messages were all delivered as they should be. The results were the same when some clients whispered and other clients sent public messages.

3. Will data be corrupted if a client disconnects at the same time that a different client connects?

   - To test this I opened multiple clients and connected all but one to the server. I clicked the disconnect button at the same time that I connected the final client to the server. Both worked as expected. the client that tried to disconnect, disconnected and the client that tried to connect was connected to the server. The list of online users displayed the correct information.

4. Is the server steady if all the clients disconnected and new clients connected afterwards?

   - To test this I opened multiple clients and connected them to the server. After a while I disconnected all of the clients and started new clients. Then I connected the new clients to the server. The server was stable during the whole process and the online users list was accurate. Thus the server was stable and working as intended.

5. Can a client receive a message while typing a message it self?

   - To test this I connected multiple clients to the server and while holding the 'a' key in the message box of the one client, I sent a message with another client. The client typing the message (holding the 'a' key) received the message without disrupting the typing of it's own message. Thus a client can indeed receive and type a message at the same time.

7 Issues encountered

No big issues were encountered. I had litle knowlede about connecting to a server via sockets and had to do a lot of research on the matter, but after that I did not encounter any major issues. The try catches gave me a hard time but it was not a major issue.

8 Significant datastructures

Sockets were interesting to me, because I have never used it before and was thus a learning experience.
Threads are also quite interesting.

9 Design

The only design decisions that I made was to make the GUI easy to read and follow. I tried not to clutter the GUI with anything that was not neccessary as to give a better user experince. The buttons are big and clearly state what they do and the text areas are big enough for the basic use of the program.

10 Compilation

In the working directory use the command 'make' to compile the program. When done with the program, use the command 'make clean' to clean up .class files.

11 Execution

Use the command 'java ProjectServer' to start the server and 'java Client_GUI <host name>' to start the client. (<host name>refers to the IP address of the machine on which the server is runnig).

12 Libraries

1. java.net.* (for sockets and server sockets)
2. java.util.Scanner
3. java.io.*
4. javax.swing.JOptionPane
5. java.awt.event.*