CSCI 2020 – SOFTWARE SYSTEMS DEVELOPMENT AND INTEGRATION

FINAL PROJECT

APRIL 19TH, 2015

LUISA ROJAS GARCIA

100518772

DESCRIPTION OF THE PROBLEM

For this project, the objecting was to create an Instant Messaging program with a Server and a Client class. The clients should be able to communicate with each other through the Server, meaning that the use of threads is necessary. The server will be receiving every message sent by any client using sockets and then it will proceed to return it to all connected clients, which will then display said message on their respective chat areas.

DISCUSSION ON THE SOFTWARE DESIGN

I thought of this program as an extension of the last assignment for this course (assignment number 5), so I worked on the code I submitted.

However, the first thing I did was design and code the GUI completely on a separate class and after, based on that design, coded the command-line part of the code on a separate class.

Once I got the command-line part working perfectly, I merged both in a third Class and got rid of the previous two – it might seem like a more complex process that it needs to be, but the concepts were clearer to me this way.

While trying to merge the two classes mentioned above, I ran into a few problems/questions, for example: should I use Object or Data input and output stream? How do I make the Connected Users field dynamic? In the end, I did some extensive research and found a way around most of them – probably not the most effective way, but the most effective way I know.

DISCUSSION ON INTEGRATED TECHNOLOGIES

In my project I integrated the following technologies (most of them covered in this course): Networking, Concurrency, Collections, Stream and File I/O, Regular Expressions, GUI (Graphical User Interface); the two latter were implemented by the use of clients, a server, sockets, threads, etc (see description of the problem).

JAVA COLLECTIONS

Initially, I was only going to implement an ArrayList where all messages would be sent and maybe outputted to a file, if the user chose to do so by using the "Download Conversation Transcript" button.

However, I also used other three (ArrayLists), one store the client's input and output streams so that the server can iterate through it if it needs to send a message to all clients, instead of one only, another one to store all clients' usernames, which is sent to every client that connects to make sure they have an updated list of the connected clients (including the ones that were connected previously). The third ArrayList corresponds to the actual clients, which was used when removing a user from the chat using the button only the server has access to.

Additionally, I used a ListModel for the JList that contained the currently connected users.

JAVA STREAM AND FILE I/O

If the user wishes to do so, they can download the conversation transcript using the download button (also available for the server). This is when the ArrayList that holds all the messages sent and received is copied into the file (.txt) and then downloaded to the user's Local Disk (C:).

REGULAR EXPRESSIONS

(This feature is only available for clients)

Before being able to access the Client GUI, the user needs to select a username. This username needs to follow the “Username123” format (A word starting with a capital letter followed by three numbers, exactly); said username cannot contain the word “Server” nor can it be longer than 44 characters, since it would distort the GUI. If said conditions are not followed, an error message will pop-up, depending on the kind of error the user may have triggered.

JAVA GUI

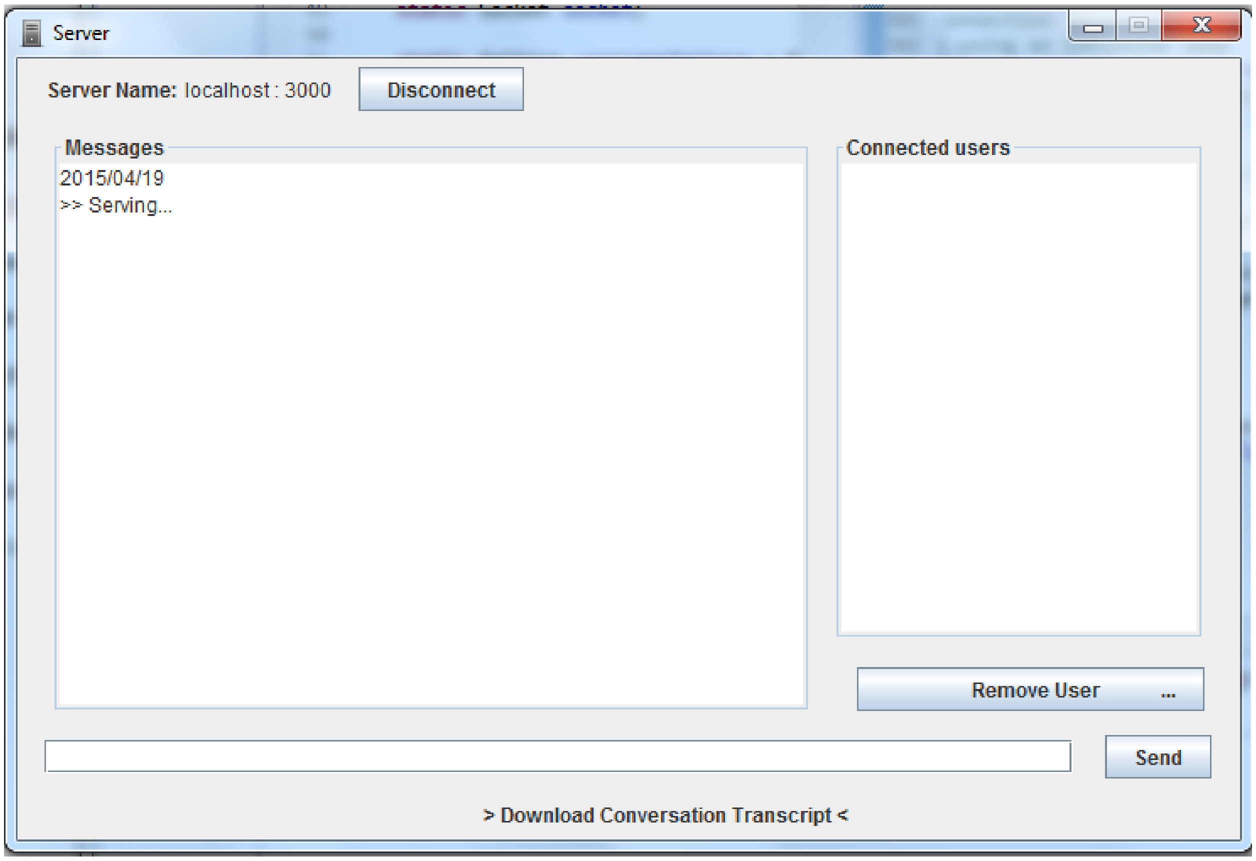
The initial design for the GUI was a lot simpler than it is now; for example, a “Connected Users” panel was added, as well as a “Remove User” button for the server and the information displayed at the top (i.e. host name, port, username).

Other than that, as described on my project proposal, they both have a chat area where all the text being sent and received is shown, as well as a user input text field, a send button and a download button.

DESCRIPTION AND EXECUTION OF THE SOFTWARE

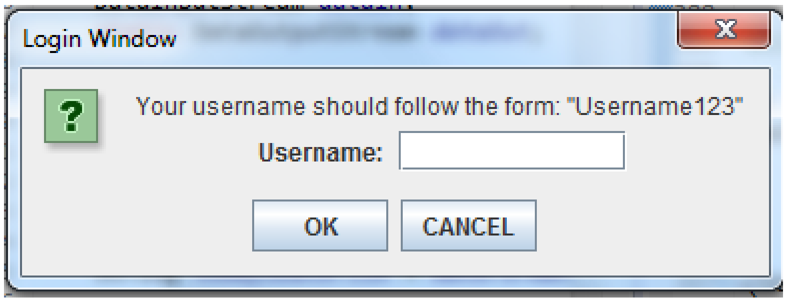
First, the Server class needs to be ran; its arguments are: host, port (int) and port (String) – These should be the same in the Client class. Today’s date should be shown at the top of the chat area and “>> Serving…” right under it (see Figure 1). Once a client has connected to the network, it will say so, as well and said client’s index number. If there is a client already connected, the server can choose to remove them from the chat using the “Remove User” button; if no clients are connected or selected from the JList, and error will pop-up. Note that the server can also download its own copy of the conversation transcript and disconnect if they choose to – in this case, clients would be notified and then disconnected as well.

After, multiple client classes can be ran. The first thing the user on the client needs to do is choose a username (see Figure 2) - see REGULAR EXPRESSIONS under DISCUSSION ON INTEGRATED TECHNOLOGIES. Once a new client joins the network (see Figure 3), everyone is notified and then they are added to the JList containing all connected users and all clients are updated (the server too). Clients can (of course) download a copy of the conversation transcript and disconnect from the network if they choose to – everyone would be notified and the disconnected client would be removed from all connected users JLists.


Server
Server Name: localhost : 3000
Disconnect
Messages
Connected users
Remove User ...
Send
> Download Conversation Transcript <

```
2015/04/19
>> Serving...
```

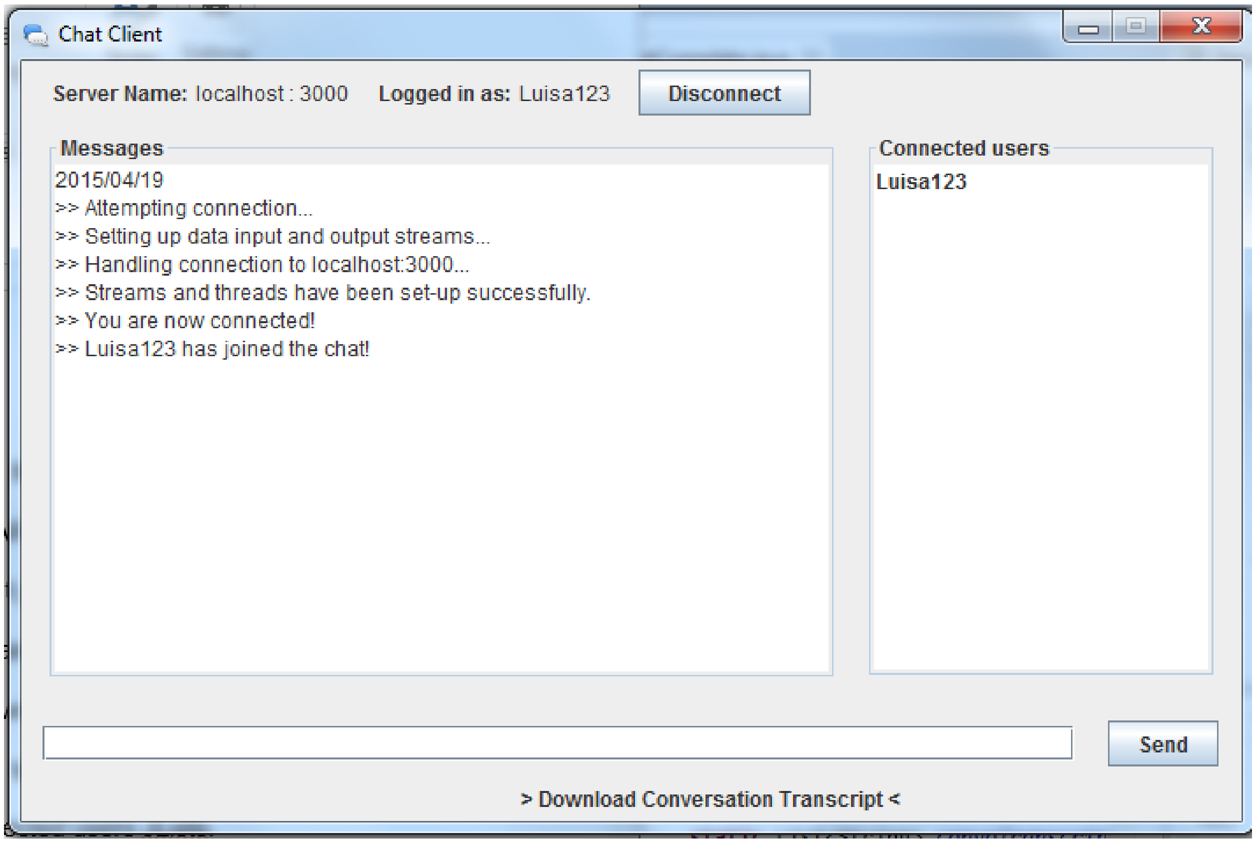
Figure 1. Server



Your username should follow the form: "Username123"

Username:

Figure 2. Client username



Server Name: localhost : 3000 Logged in as: Luisa123

```
2015/04/19
>> Attempting connection...
>> Setting up data input and output streams...
>> Handling connection to localhost:3000...
>> Streams and threads have been set-up successfully.
>> You are now connected!
>> Luisa123 has joined the chat!
```

Luisa123

Figure 3. Client

DISCUSSION ON FUTURE WORK OR POTENTIAL IMPROVEMENTS

Fortunately, I achieved all the goals I had for this project and even more; however, as I wrote the code I felt like some functionalities could have been done more effectively.