Candor

A Java Chat Server

Dylan LeMay
William Smith
Sean Gately
Andrew Lavender

Abstract

For the final project, our group decided to create a chat server in Java.We used a divide and conquer approach to make the project more manageable. For the server we built a functional text based backend. For the client we built a simple GUI for ease of use on the user. In the end we got a fully functional server/client chat program written exclusively in Java that we call Candor.

Introduction

For the final project, our group decided to create a chat server in Java. This decision was made because many of us were gamers, or at least interested in the idea of a LAN chat server. For this project, we used the simple definition of “a computer dedicated to providing the processing power to handle and maintain chatting and its users” for a chat server[2]. Anything that allows its users to sign in and chat with other users on the same server is a chat server. It does not need to have a GUI implementation or anything like that.

Also, we implemented the client side of the service as well. The client side simply allows the user to send and view received messages. Again, it does not need to implement something like a GUI to be a client.

The entirety of the server/client programming is done by sockets. Socket programming is a way to have two nodes(sockets) listen to each other. One socket forms the listener(server) while the other socket reaches out to form a connection(client). Each of the sockets must be on the same port on the network[1].

Methods

The strategy that was used for this project was a divide and conquer approach. The group was split into two small subgroups of two each: one for the text based client/server backend, and one subgroup for the GUI based client frontend.

For the backend we knew we were going to first develop a functional server that can be activated and accept connections from connecting clients. It turned out we needed to utilize sockets to make this work, and we ended up using the Java Socket library to do this. We ended up making the server side of the project activate solely through the command line. We then made the client side of the project. We decided to create the client portion in command-line first and let the frontend part of our group use that to more easily make the GUI for it.

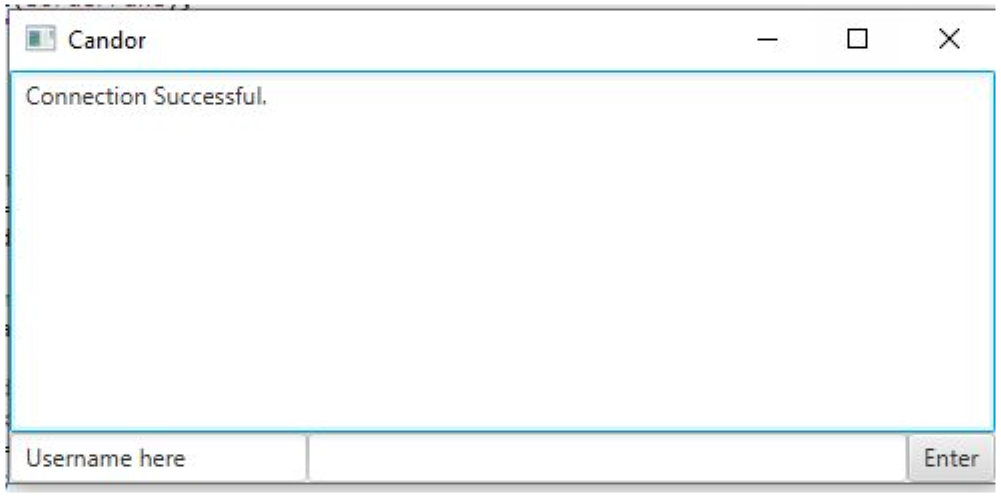
After the text based backend was developed and working as desired, a GUI implementation was added to make the product more user friendly. At first, it was thought that both server and client sides needed a GUI; however, after some deliberation, it was decided the server could stay text based. Since all it needed was a single command line argument (a positive integer for the port number) to run, there was no need to over complicate the server by adding an unnecessary GUI.

The client GUI was created in such a way that it would add to the product, yet keep a simplistic design. Only four components were added to the JavaFX scene. All components were placed on a BorderPane such that the user's current username, message, and enter button are on the bottom and the chat room is above. Once the user inputs a username and message, the enter button can be clicked to send the message to the chat room of all users. What's more, after the initial GUI window is constructed, a try/catch block is initiated to try and listen to the port number given as a command line argument.

Results

In the end, a server/client product was produced to allow users to chat with each other on the same network. A GUI was implemented for the user so that it is more appealing to the eyes and easier to use. A text area was used so the user could see all messages sent and received, two text fields were created to store the user's username and the message to be sent, and a button to send the message. These components were all put together into a simple window as seen in fig 1.

Figure 1. Candor Starting Window



```
Connection Successful.
```

Conclusions

After completing the project it is clear that frequent communication between our group members was crucial for the completion of the program. The chat server had a lot of moving parts which required knowledge from multiple domains of computer science - many of which our group was not well versed in. This required an active initiative on all of our parts to go out and study the harder to grasp concepts such as socket programming for the backend team or GUI development for the frontend team. Ultimately we are quite proud of what we have been able to create with the time allotted to us.

References

[1] “Socket Programming in C/C++.” *GeeksforGeeks*, GeeksforGeeks, 31 May 2019, www.geeksforgeeks.org/socket-programming-cc/.

[2] “What Is a Chat Server?” *Computer Hope*, Computer Hope, 7 Oct. 2019, www.computerhope.com/jargon/c/chatserv.htm.