Peer-To-Peer Connections Using IP and Ports

Tim Polizzi

Marist College

Abstract

I have created a chatroom that connects users via an IP and a port number. The project uses a GUI to connect and can support multiple users simultaneously. The project uses multiple threads in order to have the users be able to chat simultaneously and data streams in order to receive the messages and send them. The chatroom is able to connect to any user that has the GUI and knows what the host IP and port number the server is hosted on.

Introduction

During my senior year in high school, I was taught by my computer science teacher about threading and how to use data streams and used it to create an extremely simplistic chatroom that could only connect to another computer using the telnet command of terminal. Since then I have been interested in connections between computers and working with threads. When this project came up, I believed it to be the perfect time to update my old code and give it a new, sleeker look in order to make connecting simpler. My first goal of the program was to remove reliance on the telnet command, as it was outdated, and update to a more java heavy solution. Once I fixed that I decided that I would need a GUI in order to make communicating more easily for users, as terminal still had the issue of concatenating text that was being typed with text that was being received. I would have liked to further update the code to be encrypted, but time did not allow for my updates.

Detailed System Description

The system uses multiple classes in order to function as a chatroom and as a client. To first describe the class system of the chatroom itself, it first has a runnable class, ServerStarter that creates a single instance of the SimpleChat class. This class acts as the nucleus of all subsequent actions and connects a list of SimpleChatrooms to it. This list of SimpleChatrooms all are set to a key with a name attached and listed in the SimpleChat. The SimpleChatroom acts as a user handler and will connect to only a single client. It is named based on the number of prior connections to the SimpleChat, but it can be updated based on user input. This SimpleChatroom also has a single Reader and Writer method attached to it, which allow the SimpleChatroom to write to the connected user's input screen with the Writer and reads the users output to see what commands are inputted.

The next class system to go over is the user interface, which consists of a GUI and a local class structure. The local structure consists of a main class that connects to an IP at a port, and has a Reader and Writer. The writer will write data to the socket and the reader will read data from the socket and display it for the user. The GUI first requires the user to input the IP address and the port number they want to connect at, and then brings up a text area and a text field, and a button. The text area only receives output from the reader of the local structure, and the writer only uses the uses the input from the text field at the bottom once the button is pressed.

Requirements

This system is attempting to provide a simple chat structure in order to connect multiple computers, using java. It allows for communication between multiple or single people and uses a simple layout that is not confusing.

Literature Survey

There are many other programs that address the problem that I worked on, such as iMessage, Skype, Discord, and various other applications.

User Manuel

To use this system, anyone desiring to host a server will run the main method of ServerStarter and select the port they want to use. Then clients will connect using the GUI package and the local user code by inputting the server's IP and port number. Then they just need to type into the text field and click the send button. Any commands that they want to use can be used through the "/help" function that will list out available commands.

Conclusion

The goals of this system were to create a chatroom that could connect multiple clients using an easy to understand GUI. The program was also to help understand the usage of input and output streams, threading and a few other higher level class structures, all of which I believe that I completed.

References/Bibliography

Java Platform SE 7. (n.d.). Retrieved from https://docs.oracle.com/javase/7/docs/api/