Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Fall, Year: 2022), B.Sc. in CSE (Day)

Chatting Application using java and Java Swing & Socket Programming

---

Course Title: Computer Networking Lab
Course Code: CSE 312
Section:DC

Students Details

Name	ID
Zahid Hasan	201902060
-	-

Submission Date: 05/01/2022
Course Teacher's Name: Mahbubur Rahman

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

Several network systems are built to communicate with one another and are made available through service-oriented architectures. In this project, we use the client-server architecture to develop a secured Client-Server chat application. A chat application is created based on Transmission Control Protocol (TCP) where TCP is a connection-oriented protocol and in the end, multithreading is used to develop the application.

A client-server chat application consists of a Chat Client and a Chat Server and there exists a two-way communication between them. Here, Message Processor is used to interpret messages from the user, Message Interpreter is used to extract and pass the received message. Message Maker is used to constructing back the message and Client Manager is used to maintain the client's list which the sender and receiver on both sides use to interact with each other.

1.2 Motivation

A chat application makes it easy to communicate with people anywhere in the world by sending and receiving messages in real-time. We will make an app by which we can chat with each other via Social media. Allows us to send and receive messages. Chat apps make it easy, simple, and fast to communicate with someone and it's also easy to use. [1].

1.3 Problem Definition

1.3.1 Problem Statement

This project is to create a chat application with a server and users to enable the users to chat with each others. To develop an instant messaging solution to enable users to seamlessly communicate with each other. The project should be very easy to use enabling even a novice person to use it.

1.3.2 Complex Engineering Problem

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	About socket programming and threading
P2: Range of conflicting requirements	The conflicting requirement was at the as same time sending a text in multiple users but we overcome that
P3: Depth of analysis required	Depth analysis was required to do the experiment when the user was sending the message how the receiver is getting.
P4: Familiarity of issues	we learn a lot of things for this project which our teacher learn and some extra things we learn from self.

1.4 Design Goals/Objectives

The following project concerns the development of a multi-threaded chat/messaging platform in java, with a thorough definition and characterization. Here we will discuss both Server-side programming and Client-side programming with proper coding and their outcome. It is presumed in socket programming that the client sends some information first and then the server or other clients respond to that information. This does not often be the case in realistic contexts, it is not needed to deliver a message to someone to be able to receive one. Whenever it is sent, a recipient can readily receive a response, whereas transmitting and receiving must be introduced as independent rather than sequential processes. [2].

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

Client-server model is the standard model which has been accepted by many for developing secure network applications. In this model, there is a notion of client and a notion of the server. A chat application is basically a combination of two applications:

- Server application
- Client application.

The server application runs on the server computer and client application runs on the client computer (or the machine with the server). In this chat application, a client can send data to anyone who is connected to the server

2.2 Implementation

Here is an example of how a very simple client-server chat application works. These are the stages involved:

Step 1: In any Client/Server Application, we need to run the server before the client, because the server keeps waiting for the client to be connected.

Step 2: Server keeps listening for the client on an assigned IP & Port

Step 3: For establishing connection client must know the IP & Port of the server.

Step 4: When we start Client Application, It creates a connection to the server.

Step 5: After the Successful connection Client & Server Applications can send & receive messages. [3].

Server Side:

For the Server application, we use the Server Socket Class that binds to a specific port number. We then create a server socket and listen for a connection with the Client and accept it. The thread Server Thread is instantiated and started. All the threads are added to an Array List so that multiple clients can send messages to each other.

In ServerThread.java, we received sockets and a list of active threads from the Main.java using constructor. When we start the thread from main, the run method is called. The BuffereReader helps us to receive information from the client. Any information that Server wants to send is sent using PrintWriter. The method printToALL-Clients() sends the output to each client in the thread.

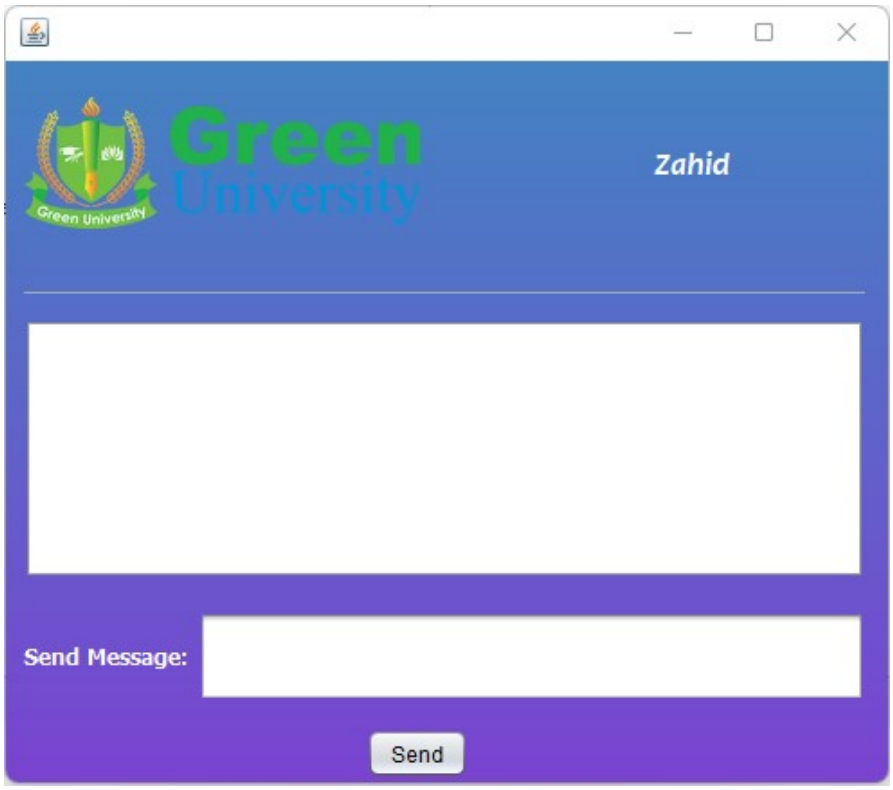

Figure 2.1: Server side

Client Side:

For the Client, we use the Socket class and initiate the connection to a server bypassing the IP address and port number. We use the Scanner to get the input from the user and send the data to the server using the PrintWriter object.

We have used ClientThread class to listen to the response from the server, without getting blocked while reading from a Scanner. We have used input BufferReader to get information from the client.

Figure 2.2: Client side

2.3 Algorithms

Algorithm for implementing this chat application based on client-server model is

Server

Begin

Initiate server process listening for requests for all time

Receive request

Create a thread for client

Create Socket connection with client thread

While Comm. Not Ends

Do input/output streams

Close streams

Close socket

Kill thread

End

Client

Begin

Initiate client process

Create a thread for making request

Make Request

Create client thread for Data Transfer

While Comm. Not Ends

Do input/output streams

Close streams

Close socket

Kill thread

End

Chapter 3

Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

Sockets Programming helps us to communicate with the various computers running on a network. In Java, Socket programming can be either connection-oriented or connectionless. We will design the connection-oriented application that uses the Client-Server model.

In the Client-Server model, the Server has a unique IP Address and port number. The client tries to make a connection with the server using this port number and IP address. The server listens and accepts the connection. Once the connection is built, the Server can receive a message from the client as well as respond back a message to the client.

Since the project involves multiple clients that can send messages to each other, we will use threads. The thread ensures that every client gets their own server socket.

3.2 Results Overall Discussion

This is our final result and we smoothly message with server to client without any error.

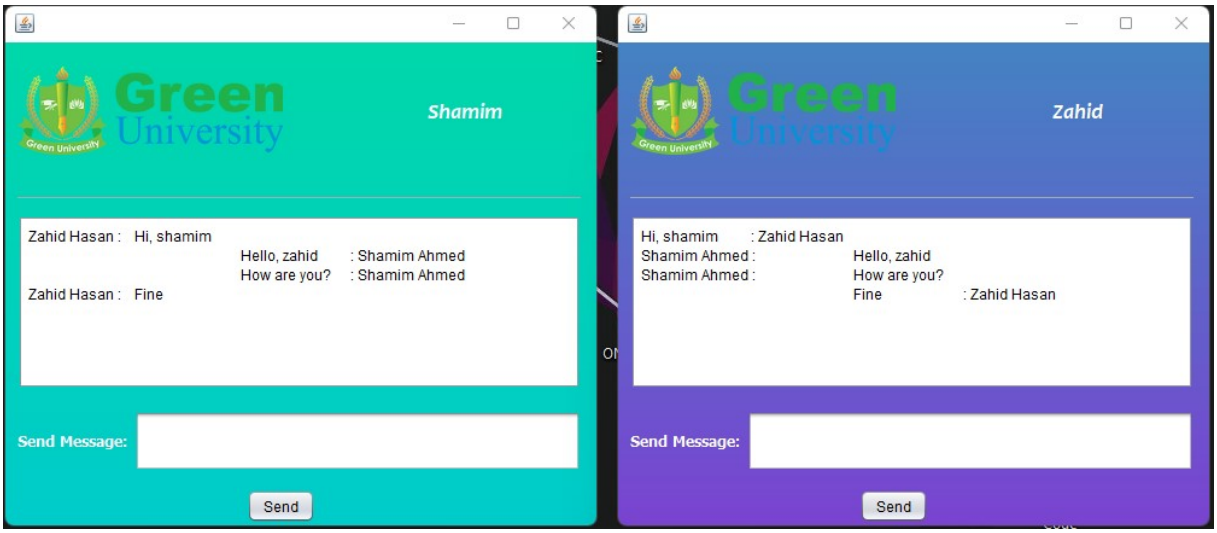

Figure 3.1: Server to Client side messaging

Chapter 4

Conclusion

4.1 Discussion

This project is to create a chat application with a server and users to enable the users to chat with each others. To develop an instant messaging solution to enable users to seamlessly communicate with each other. The project should be very easy to use enabling even a novice person to use it.

4.2 Limitations

In this project, we can not chat with multiple clients and we can not share another file share to our chat application. this limitation we will be solved this in our newer future works.

4.3 Scope of Future Work

we will try to add our project some more features which are -

- Implement multi-clients functionality;
- Enable clients to communicate with each other;

References

[1] Omid C Farokhzad and Robert Langer. Impact of nanotechnology on drug delivery. *ACS nano*, 3(1):16–20, 2009.

[2] Uthayasankar Sivarajah, Muhammad Mustafa Kamal, Zahir Irani, and Vishanth Weerakkody. Critical analysis of big data challenges and analytical methods. *Journal of Business Research*, 70:263–286, 2017.

[3] Douglas Laney. 3d data management: controlling data volume, velocity and variety. gartner, 2001.