Green University of Bangladesh

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

Green Chat (Customer Manager Chatting Application)

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

Student Details

Name	ID
Hamad Ismail	201902046

Submission Date: 07/01/2023

Teacher's Name: Rusmita Halim Chaity

Lab Report Status	
Marks:	Signature: .....
Comments:	Date: .....

Contents

1 Introduction **4**

1.1 Overview 4

1.2 Objective 4

1.3 Problem Definition 5

1.3.1 Problem Statement 5

1.3.2 Complex Engineering Problem 5

2 Design **6**

2.1 Introduction 6

2.2 Project Details 6

2.2.1 Server & Client Connection 6

2.2.2 Graphical User Interface 6

2.3 Implementation 7

2.3.1 Establishing The Connection 7

2.3.2 User Interface Designing 8

2.4 Algorithms 13

3 Performance Evaluation **14**

3.1 Simulation Procedure 14

3.2 Results Analysis 15

3.2.1 Working Perspective 15

3.2.2 Simulation Perspective 15

3.3 Results Overall Discussion 15

3.3.1 Complex Engineering Problem Discussion 15

4 Conclusion **16**

4.1 Introduction 16

4.2 Scope of Future Work 16

4.3 Contribution 16

References

Chapter 1

Introduction

1.1 Overview

Chatting is a method of using technology to adding people and ideas together despite of the geographical barriers. The technology has been available for years but the acceptance it was quit recent. Our project is an example of a multiple client chat server.
Creating server with multiple client was intermediate level work for us. But working with java swing was the new task for us. For learning java swing we use some website those website helping us to serve the code from our given design.

1.2 Objective

Goal to create a chat application using socket programming with java swing. From here multiple customers can send message to the manager and manager can reply all of the customers but one customer never ever can seen other customers message. We give the User Interface for the project just like a telegram application.

By using this method here already have many chat application. From this chat application we can get the idea about all other chat apps. We use whats app, telegram, messenger or many other chat application for personal chatting or group chatting. All these application runs depend on the socket programming.

1.3 Problem Definition

1.3.1 Problem Statement

Creating server with multiple client was intermediate level work for us. But working with java swing was the new task for us. For learning java swing we use some website those website helping us to serve the code from our given design.

1.3.2 Complex Engineering Problem

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

Name of the WP Attributes	**Explain how to address**
WP1: Depth of knowledge required	About socket programming should have the proper knowledge
WP2: Range of conflicting requirements	The conflicting requirement was at as same time sending text in multiple user but we over come from that
WP3: Depth of analysis required	Depth analysis was required to done the experiment when user was sending the message how the receiver is getting
WP4: Extent of applicable codes	This code can be applied for group Chat
WP5: Extent of stakeholder involvement and conflicting requirements	There requirement was to a better user interface and a secure chatting app
WP6: Interdependence	Though we done this project with the concept of socket programming but java swing concept also was require

Chapter 2

Design

2.1 Introduction

We have build this project using IntelliJ IDEA IDE.

2.2 Project Details

2.2.1 Server & Client Connection

For connecting all users who are actually the client with server we use server socket at server side and socket at client(user) side. This sockets connect them together with a port number.

2.2.2 Graphical User Interface

For making user interface we have use Java Swing technology. Swing is a lightweight Java graphical user interface (GUI) that is used to create various applications. Swing has platform-independent components. It enables the user to create buttons and scroll bars. Swing includes packages for creating desktop applications in Java.

Here we will describe some of the basic component of java Swing.

Component	Description
JLabel	An area where uneditable text or icons can be displayed.
JTextField	An area in which the user inputs data from the keyboard. The area can also display information.
JButton	An area that triggers an event when clicked.
JCheckBox	A GUI component that is either selected or not selected.
JComboBox	A drop-down list of items from which the user can make a selection by clicking an item in the list or possibly by typing into the box.
JList	An area where a list of items is displayed from which the user can make a selection by clicking once on any element in the list. Double-clicking an element in the list generates an action event. Multiple elements can be selected.
JPanel	A container in which components can be placed.

Figure 2.1: Some Basic GUI Components

2.3 Implementation

2.3.1 Establishing The Connection

Here we work for the connection between server and client and handle multiple client.

Server Side

```
public static void main(String[] args) throws Exception {
    ServerSocket s = new ServerSocket(2023);
    while(true) {
        Socket socket = s.accept();
        Server server = new Server(socket);
        DataInputStream din = new DataInputStream(s.getInputStream());
        dout = new DataOutputStream(s.getOutputStream());
    }
```

User Side

```
try {
    Socket socket = new Socket("localhost", 2023);
    writer = new BufferedWriter(new OutputStreamWriter(socket.
       getOutputStream()));
    reader = new BufferedReader(new InputStreamReader(socket.
       getInputStream()));
    } catch (Exception e) {
        e.printStackTrace();
    }
```

Client Handler

```
    Thread new_tunnel = new ClientHandler(s, din, dout);
    new_tunnel.start();

    public ClientHandler(Socket comTunnel, DataInputStream disTunnel,
       DataOutputStream dosTunnel)
    {
        this.com_tunnel = comTunnel;
        this.dis_tunnel = disTunnel;
        this.dos_tunnel = dosTunnel;
    }

    public void run () {
    try {
        while(true) {
            String msg = dis_tunnel.readUTF();
            JPanel panel = Server.formatLabel(msg);

            JPanel left = new JPanel(new BorderLayout());
            left.add(panel, BorderLayout.LINE_START);
            Server.vertical.add(left);
            Server.f.validate();
        }
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}
```

2.3.2 User Interface Designing

Title bar color

```
    f.setLayout(null);
    JPanel p1 = new JPanel();
    p1.setBackground(new Color(42, 171, 238));
    p1.setBounds(0, 0, 450, 70);
    p1.setLayout(null);
    f.add(p1);
```

Back button

```
    //back button png image paste here
    ImageIcon i1 = new ImageIcon(ClassLoader.getSystemResource("icons
       /3.png"));
    Image i2 = i1.getImage().getScaledInstance(25, 25, Image.
       SCALE_DEFAULT);
    ImageIcon i3 = new ImageIcon(i2);
    JLabel back = new JLabel(i3);
    back.setBounds(5, 20, 25, 25);
    p1.add(back);
    //close the interface while pressing the back png
    back.addMouseListener(new MouseAdapter() {
        public void mouseClicked(MouseEvent ae) {
            System.exit(0);
        }
    });
```

Image Icon

```
//group icon here
    ImageIcon i4 = new ImageIcon(ClassLoader.getSystemResource("icons/
       hamad.png"));
    Image i5 = i4.getImage().getScaledInstance(60,60, Image.
       SCALE_DEFAULT);
    ImageIcon i6 = new ImageIcon(i5);
    JLabel profile = new JLabel(i6);
    profile.setBounds(40, 5, 60, 60);
    p1.add(profile);
```

User Name & User Profile

```
    //group name with placement and font style with size
    JLabel name = new JLabel("Hamad (Manager)");
    name.setBounds(110, 15, 100, 18);
    name.setForeground(Color.WHITE);
    name.setFont(new Font("SAN_SERIF", Font.BOLD, 18));
    p1.add(name);
    //group members name with placement and font with style
    JLabel status = new JLabel("Hamad (Customer)");
    status.setBounds(110, 35, 160, 18);
    status.setForeground(Color.WHITE);
    status.setFont(new Font("SAN_SERIF", Font.BOLD, 14));
    p1.add(status);
```

Video, Audio & Option Icon

```
    //video call icon
    ImageIcon i7 = new ImageIcon(ClassLoader.getSystemResource("icons/
       video.png"));
    Image i8 = i7.getImage().getScaledInstance(30, 30, Image.
       SCALE_DEFAULT);
    ImageIcon i9 = new ImageIcon(i8);
    JLabel video = new JLabel(i9);
    video.setBounds(300, 20, 30, 30);
    p1.add(video);
    //audio call icon
    ImageIcon i10 = new ImageIcon(ClassLoader.getSystemResource("icons/
       phone.png"));
    Image i11 = i10.getImage().getScaledInstance(35, 30, Image.
       SCALE_DEFAULT);
    ImageIcon i12 = new ImageIcon(i11);
    JLabel phone = new JLabel(i12);
    phone.setBounds(360, 20, 35, 30);
    p1.add(phone);
    //option icon here with 3 dots
    ImageIcon i13 = new ImageIcon(ClassLoader.getSystemResource("icons
       /3icon.png"));
    Image i14 = i13.getImage().getScaledInstance(10, 25, Image.
       SCALE_DEFAULT);
    ImageIcon i15 = new ImageIcon(i14);
    JLabel morevert = new JLabel(i15);
    morevert.setBounds(420, 20, 10, 25);
    p1.add(morevert);
```

Designing Text Panel

```
public static JPanel formatLabel(String out) {
    JLabel output = new JLabel("<html><p style=\"width: 150px\">" + out
        + "</p></html>");
    output.setFont(new Font("Tahoma", Font.PLAIN, 16));
    output.setBackground(new Color(0, 255, 255));
    output.setOpaque(true);
    output.setBorder(new EmptyBorder(0, 15, 0, 50));
    panel.add(output);

    Calendar cal = Calendar.getInstance();
    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm");

    JLabel time = new JLabel();
    time.setText(sdf.format(cal.getTime()));
```

```
    panel.add(time);
    return panel;
}
```

Working Text Panel

```
public void actionPerformed(ActionEvent ae) {
try {
    String out = text.getText();
    JPanel p2 = formatLabel(out);

    a1.setLayout(new BorderLayout());

    JPanel right = new JPanel(new BorderLayout());
    right.add(p2, BorderLayout.LINE_END);
    vertical.add(right);
    vertical.add(Box.createVerticalStrut(15));

    a1.add(vertical, BorderLayout.PAGE_START);

    dout.writeUTF(out);
    text.setText("");

    f.repaint();
    f.invalidate();
    f.validate();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

The workflow

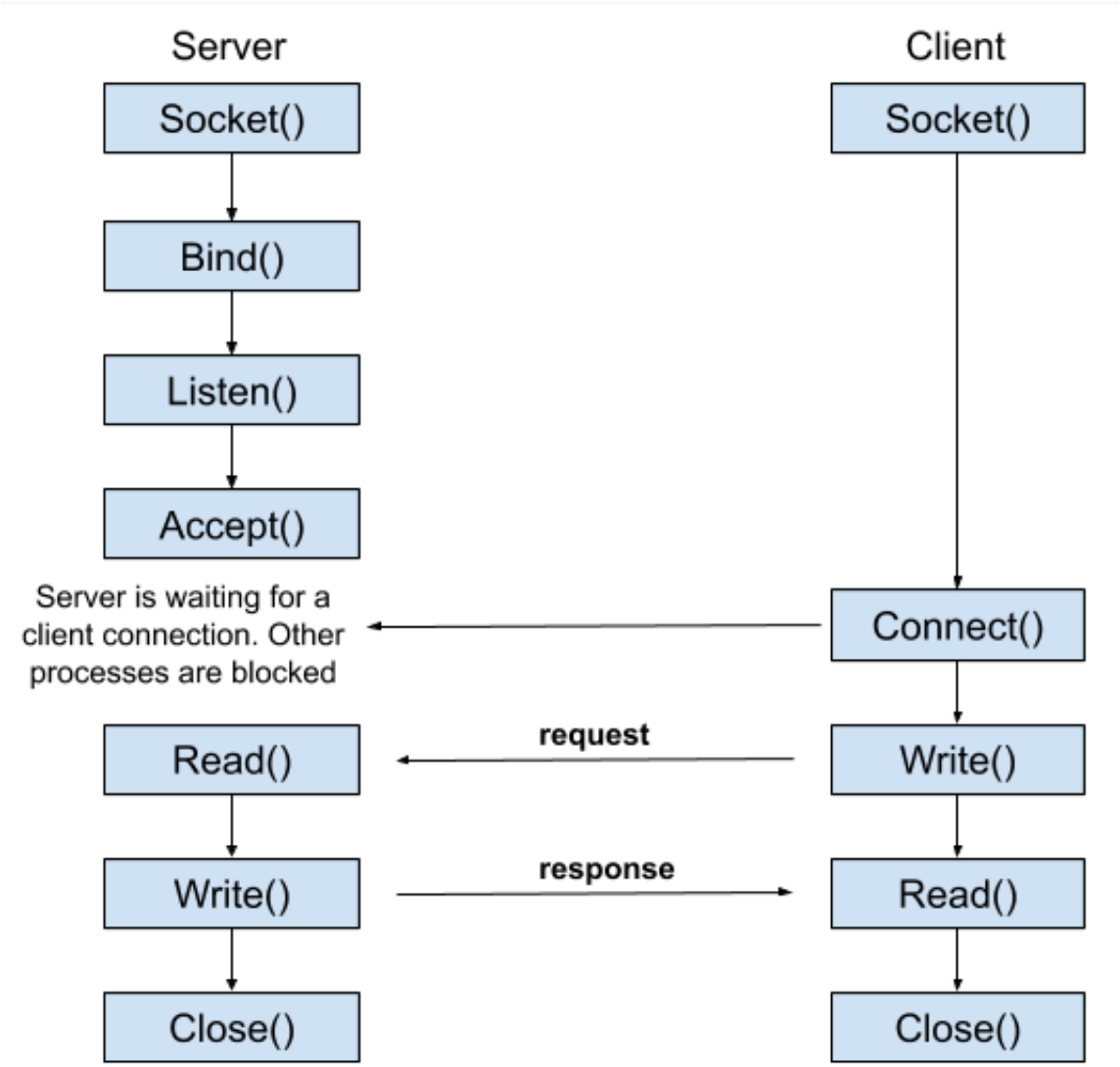

Figure 2.2: Server- client Connection Flow

Tools and libraries

We use IntelliJ IDEA IDE [1] for build and run this project. We use [2] , [3] & [4] open libraries for solving the errors.

2.4 Algorithms

Algorithm 1: Algorithm of Server Side Socket Connection

```
1  Begin /* Initiate server process listening for requests for
       all time                                                  */
2  if Receive request then
3      Create a thread for client Create Socket connection with client thread
4      while Com != Ends do
5          Input/Output Stream
6          close streams
7          close socket
8          kill thread
9      close socket
10 End
```

Algorithm 2: Algorithm of Client Side Socket Connection

```
1  Begin /* Initiate client process                             */
2  Create a thread for making request
3  Make Request
4  Create client thread for Data Transfer
5  while Comm. Not Ends do
6      input/output streams
7  Close streams
8  Close socket
9  Kill thread
10 End
```

Chapter 3

Performance Evaluation

3.1 Simulation Procedure

We just need IntelliJ IDEA IDE for run this project.

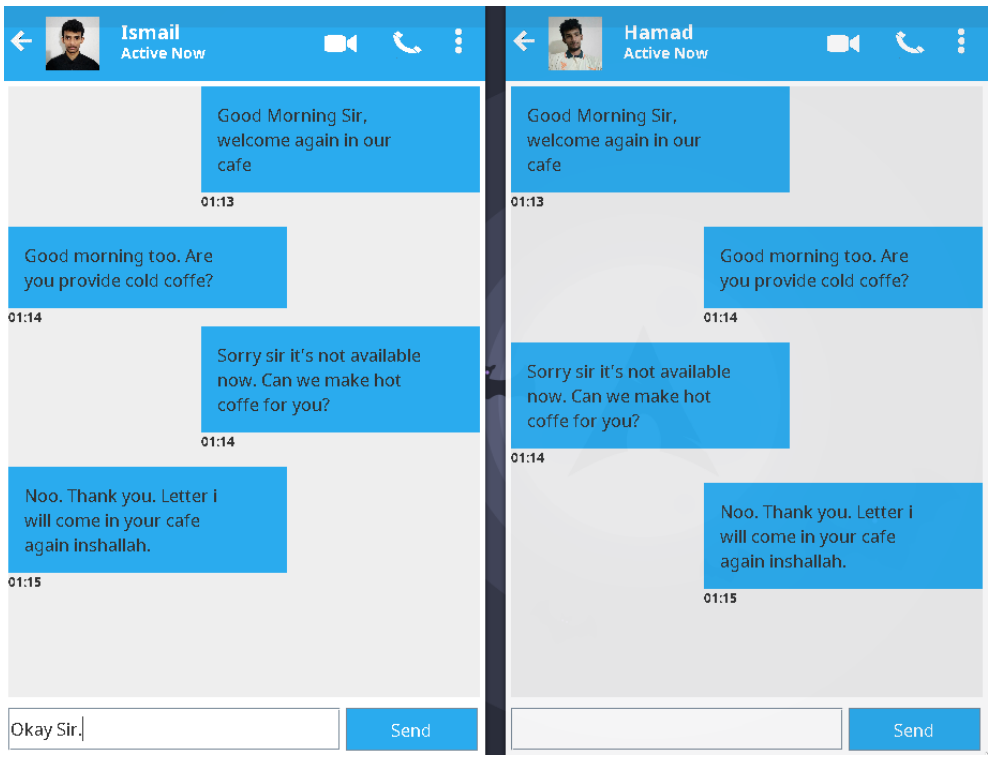

Figure 3.1: Customers & Manager View

3.2 Results Analysis

3.2.1 Working Perspective

While users customers send message to the manager the manager successfully viewing the message and can reply.
Initial connection and every users connection working successfully. While users clicking the back button then automatically users gets disconnect from the server as defined in implementation.

3.2.2 Simulation Perspective

In simulation, we notice that only manager can view customers message other user can't be read. But a manager can serve one or more customers at the same time.

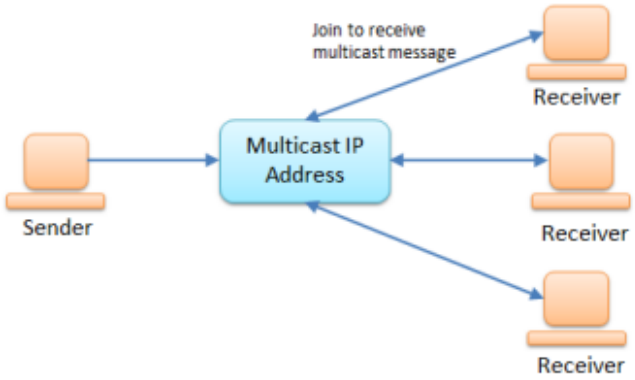

Figure 3.2: Handle Multiple Customers

3.3 Results Overall Discussion

A general discussion about how your result has arrived should be included in this chapter. Where the problems detected from your results should be included as well.

3.3.1 Complex Engineering Problem Discussion

[Optional] In this subsection, you can discuss in details the attributes that have been touched by your project problem in details. This has already been mentioned in the Table 1.1.

Chapter 4

Conclusion

4.1 Introduction

We Developed network applications in Java by using sockets, threads, and swing. These software is portable, efficient, and easily maintainable. Our developed chatting software is unique in its features and more importantly easily customizable. The java.net package provides a powerful and flexible set of classes for implementing network applications. Typically, programs running on client machines make requests to programs on a server Machine. These involve networking services provided by the transport layer. The most widely used transport protocols on the Internet are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

4.2 Scope of Future Work

Files transfer: this will enable the user to send files of different formats to others via the chat application.
Voice chat: this will enhance the application to a higher level where communication will be possible via voice calling as in telephone.
Video chat: this will further enhance the feature of calling into video communication

4.3 Contribution

Hamad Ismail Server Side & User interface.

References

[1] I. IntelliJ, "the most intelligent java ide," *JetBrains [online].[cit. 2016-02-23]. Dostupné https://www.jetbrains.com/idea/ chooseYourEdition*, 2011.

[2] "Latex Design latex error finder." https://tex.stackexchange.com. Accessed Date: 2023-01-03.

[3] "Code Helper error solving." https://stackoverflow.com. Accessed Date: 2023-01-03.

[4] "Geek For Geeks theory helper." https://www.geeksforgeeks.org. Accessed Date: 2023-01-03.