TERM PROJECT

On

Chat Application using concepts of Object Oriented Methodology

Submitted By :

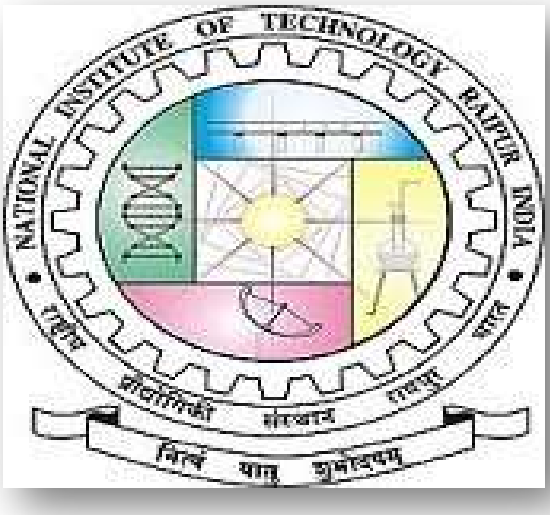
VEDANSH TANDON

Roll No : 20118111

3rd Semester

Information Technology

National Institute of Technology, Raipur



Under the Supervision of

Dr. Ashish Mishal

National Institute of Technology, Raipur

Acknowledgement

No work of any significance can de done and under taken to ones satisfaction without the help and guidance of others. Therefore, I would like to thank Dr. Ashish Mishal sir for his proficient whole hearted help and guidance to me in making this " Chat Application project using the concepts of Object Oriented Methodology ." I am highly grateful and thankful to you sir for your help and guidance.

Vedansh Tandon

Roll No: 20118111

3rd Semester

Information Technology

National Institute of Technology, Raipur

Abstract

In this Term Project, I made a Chatting Application by using the fundamental concepts of Object Oriented Methodology. This Chatting Application has been developed with the help of JAVA programming language. The solo objective of making this chatting application is to develop a network between two users sitting at any two corners of the world so that they can communicate, thereby making distant communication more easier. Moreover, the fundamental concepts of Object Oriented Methodology like Socket Programming, file handling, exception handling and Graphical User Interface (GUI) has also been demonstrated effectively.

Basically, this Chatting Application is a desktop based application in which the GUI part was added with the help of JAVA Swing foundation classes. The network established between the server and the client is with the help of network or socket programming in java. Moreover, the Calendar class has also been used to display the time at which the message was sent. Hence, with all the functions and concepts working flawlessly we can say definitely say that on further upgradation and modification it can to launched into market for general Chatting purpose.

Introduction

Chatting or Tele-Conferencing is a method of using technology to bring people and ideas together despite of the geographical barriers. The motive of making this text chatting application is to make communication more easy and also to learn the fundamental concepts of Object Oriented Methodology. This desktop based chatting application first establishes a connection (network) between two users present at any two corners of the world and then with the help of socket programming they communicate. Moreover , to make this application more appealing I have added GUI to it with the help of Java swing classes which will be adding additional features to my project. To have a back-ups of the chats between the two users I have created a file named 'Chat.txt' which will be storing the Chats of the two users. The Following are the major Concepts of Object Oriented Methodology which I have implemented in my project Chat Application :

- **Socket Programming / Network Programming** : Socket Programming is used for communication between the applications that are running on different Java Run-time Environment. It can be either connection-oriented or connectionless. As a whole, a socket is a way to establish a connection or a network between a client and a server.
- **GUI using Java Swing** : Java Swing tutorial is a part of Java Foundation Classes that are used to create window-based applications. The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.
- **File Handling** : Using files, a program can store data on a storage device. To perform various actions on a file, such as read, write, and so on, file handling is required. Here, Chat.txt file store the data of chats between the two users.
- **Exception Handling :** The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained. In my project I have used exception handling effectively to avoid run-time errors and to avoid abrupt ending of my application.

Road-Map :

Step 1 : Create two java classes named **Server.java** and **Client.java**

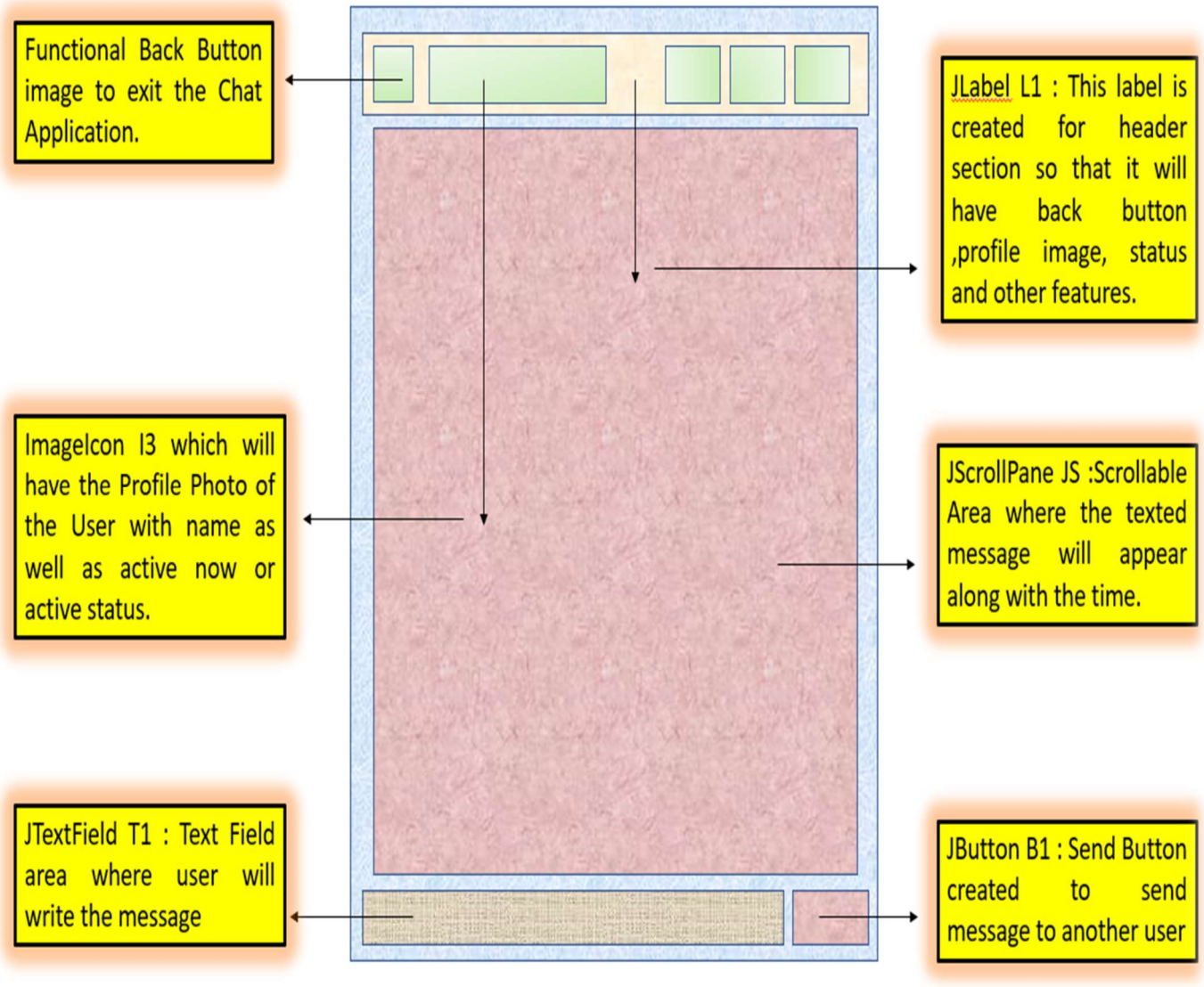
Step 2 : Create a Layout of the Chatting Application using Graphical User Interface with the help of Java Swing.

Step 3 : With the help a socket programming establish a connection between the server and client class.

Step 4: Use the concept of File Handling, create a file named **Chat.txt** to store the chants between the two users.

Step 5: Adding additional features to this Chatting Application.

Designing the Layout of Chat Application:



Purpose of importing javax.swing :

Java Swing provides the following tools with the help of which we can add GUI and functionality to our desktop based application.

Such tools are used in this project from javax.swing package :

- **JButton** : The class JButton is an implementation of a push button. This component has a label and generates an event when pressed. It can also have an Image.
- **JTextField** : JTextField is a lightweight component that allows the editing of a single line of text.
- **JScrollPane** : A JScrollPane provides a scrollable view of a component. When screen real estate is limited, use a scroll pane to display a component that is large or one whose size can change dynamically.
- **JLabel** : JLabel is used to display a short string or an image icon. JLabel can display text, image or both .
- **JPanel** : The JPanel is a simplest container class. It provides space in which an application can attach any other component. It inherits the JComponents class.
- **AddKeyListener(new KeyAdapter())** : The listener interface for receiving keyboard events (keystrokes). ... A keyboard event is generated when a key is pressed, released, or typed.
- **setBackground( )** : in-built function in swing class to change background color of the JPanel, JLabel, JTextField or JFrame.
- **setForegound( )**: in-built function in swing class to change foreground color of the JPanel, JLabel, JTextField or JFrame.
- **setBounds( )**: in-built function in swing class to set the location of the JPanel, JLabel, JTextField or JFrame.
- **setBorder( )**: in-built unction in swing class to set border settings of the JPanel, JLabel, JTextField or JFrame.

  .....And many others inbuilt functionality.

Purpose of Importing Calendar package :

Java also provides Calendar and SimpleDateFormat Package from which we can find out the current date and time of the system. With the help of inbuilt functionality offered by this pre-defined package I have displayed the time at which the message was sent from Server to Client or vice versa.

Purpose of Importing IO Package :

Java provides java.io.* package .This package provides help in system input and output through data streams, serialization and the file system.

With the help of this package operations like writing on file 'Chat.txt' , transferring data from DataInputStream and DataOutputStream were performed.

Concept Of Socket Programming :

Socket programming is a means of communicating data between two computers across a network. Connections can be made using either a connection-oriented protocol or a connectionless protocol. In our case, we will use TCP/IP which is a connection-oriented protocol. Before exchanging data, computers must establish a link that is for connection-oriented protocols.

So in this project I have created two classes Server class and Client class. Later, with the help of socket programming I accepted the request of Client class and have established a connection between them. Later ,with the help of DataInputStream and DataOutputStream I have communicated the messages between two users thereby making this Chat Application a successful and a flawless project.

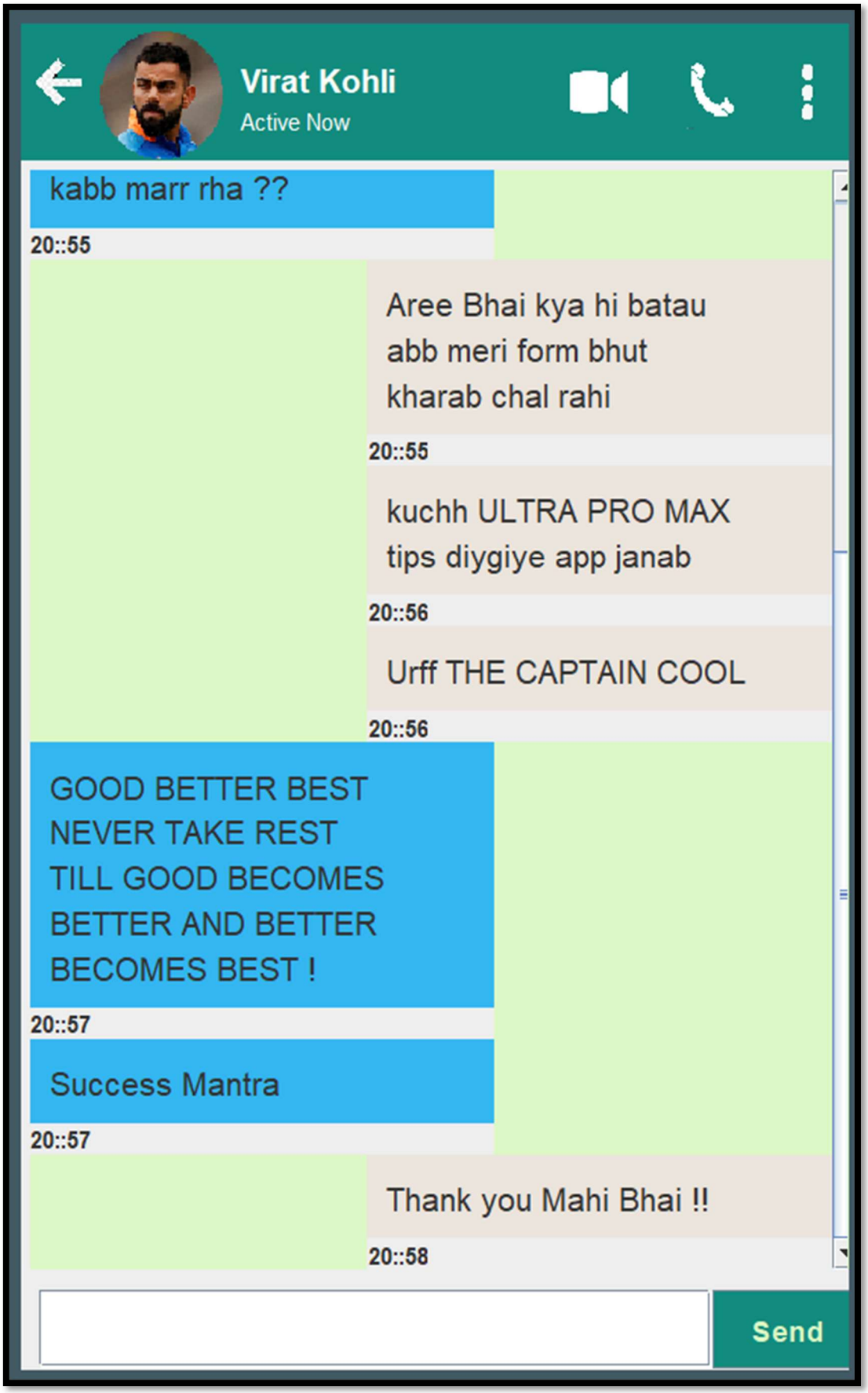
Concept Of File Handling :

File Handling can be defined as reading and writing to a file. The java.io package allows us to work different formats of files and gives the liberty to user to read, write ,append ,delete and update data of a file. In this Chat Application Project I have used the concept of file handling to write the chats data on a file named 'Chat.txt' and hence it would act as a backup to restore the chats between the two users.

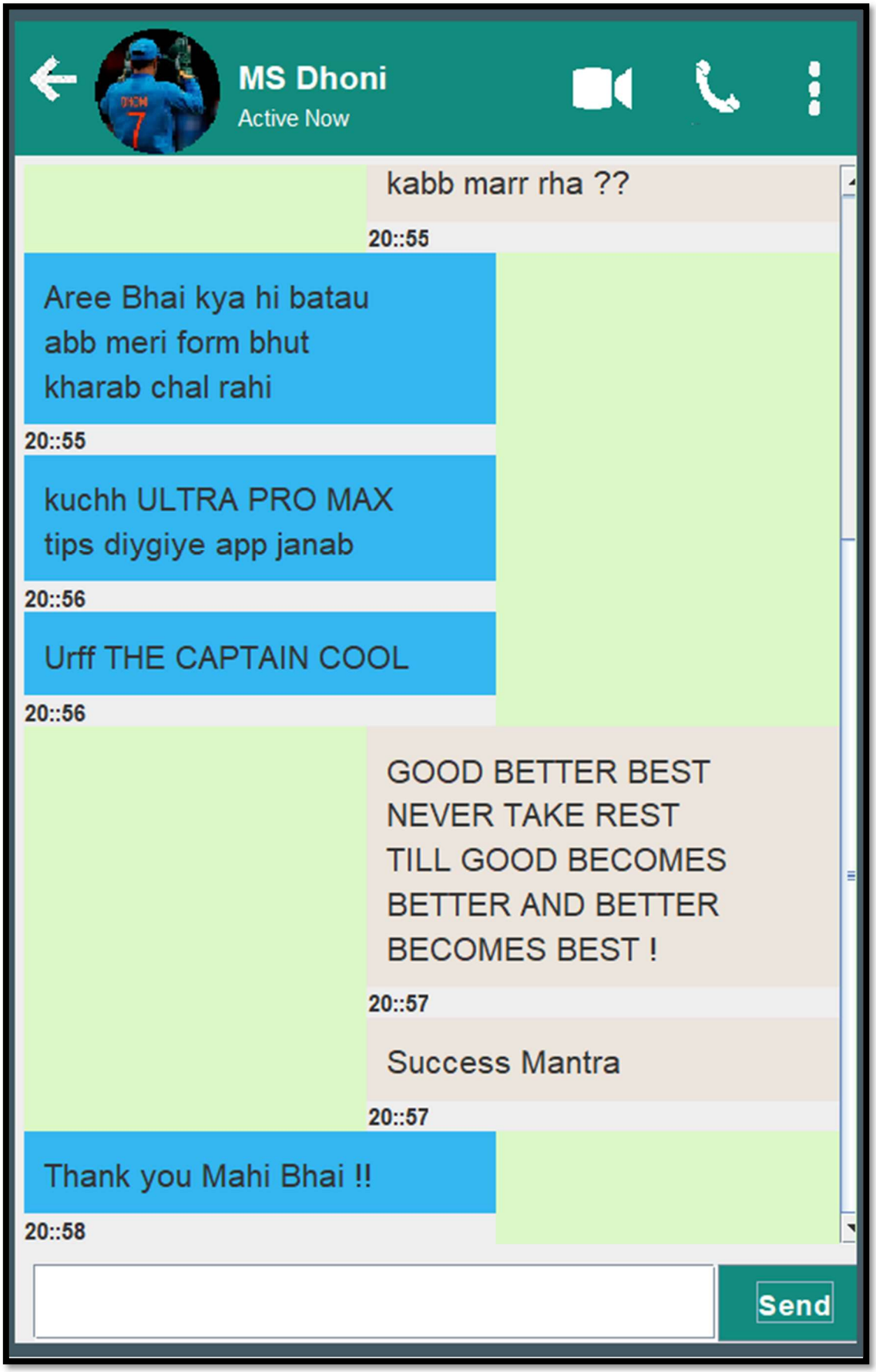
Concept Of Exception Handling :

The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained. In my project I have used exception handling while performing file Handling and socket programming because these were the only two sections in my code where an exception may arise. Hence the concept of exception handling was also implements in my code using try, catch, throw, throws and throwable keywords.

Final Server Image After All Coding and Designing :



Final Client Image After All Coding and Designing :



Server Class Code :

```
// Packages imported in CHAT APPLICATION PROJECT
// For GUI , TIME MANAGEMENT, INPUT_OUTPUT and FILE HANDLING

import javax.swing.*;
import javax.swing.border.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.io.*;
import java.util.Calendar;
import java.text.SimpleDateFormat;

// CLASSS SERVER
public class Server extends JFrame implements ActionListener {

    // P1 is the Panel created for HEADER SECTION which contains DP, Active
status, other images
    // T1 is the TEXT_FIELD used to create the MESSAGE_TYPING text area
    // B1 Button is created to send message
    // A1 Panel is created To create a area where messages will Appear
    JPanel P1;
    JTextField T1;
    JButton B1;
    static JPanel A1;
    static Box vertical = Box.createVerticalBox();
    //used for socket programming
    static ServerSocket skt;
    static Socket s;
    // used for data transfer
    static DataInputStream din;
    static DataOutputStream dout;
    // TYPING variable used to check whether the current person is typing or
not
    Boolean typing;
    static JFrame F1 = new JFrame();

    // all coding part of Jframe is done in the constructor
    Server() {

        F1.setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        // HEADER SECTION
        // creating a panel P1 to show the back button DP ,videocall, image,
and settings buttons and options
        P1 = new JPanel();
        P1.setLayout(null);
        P1.setBackground(new Color(18, 140, 126));
        P1.setBounds(0, 0, 450, 70);
        F1.add(P1);

        // BACK BUTTON IMAGE AND FUNCTIONALITY
```

```
        // getSystemResources -> function to take resources like images and
files from system
        ImageIcon I1 = new
ImageIcon(ClassLoader.getSystemResource("chatting/application/icons/3.png"));
        Image I2 = I1.getImage().getScaledInstance(30, 30,
Image.SCALE_DEFAULT);
        ImageIcon I3 = new ImageIcon(I2);
        JLabel L1 = new JLabel(I3);
        L1.setBounds(5, 15, 30, 30);
        P1.add(L1);
        // adding FUNCTIONALITY to BACK BUTTON IMAGE
        L1.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent ae) {
                //if clicked then exit the system SYSTEM_CALL
                System.exit(0);
            }
        });

        //adding DP - PROFILE IMAGE
        // setting the DP image of VIRAT_KOHLI
        ImageIcon I4 = new
ImageIcon(ClassLoader.getSystemResource("chatting/application/icons/virat-
modified.png"));
        Image I5 = I4.getImage().getScaledInstance(65, 65,
Image.SCALE_DEFAULT);
        ImageIcon I6 = new ImageIcon(I5);
        JLabel L2 = new JLabel(I6);
        L2.setBounds(42, 4, 65, 65);
        P1.add(L2);

        //writing VIRAT KHOLI NAME on the panel
        Label L3 = new Label("Virat Kohli");
        L3.setFont(new Font("SAN_SERIF", Font.BOLD, 17));
        L3.setBounds(115, 20, 120, 20);
        L3.setForeground(Color.WHITE);
        P1.add(L3);

        // adding ACTIVE_NOW STATUS
        Label L4 = new Label("Active Now");
        L4.setForeground(Color.WHITE);
        L4.setFont(new Font("SAN_SERIF", Font.PLAIN, 12));
        L4.setBounds(115, 40, 120, 20);
        P1.add(L4);
        //using timer function for changing typing status
        Timer t = new Timer(1, new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                if (!typing) {
                    L4.setText("Active Now");
                }
            }
        });
        t.setInitialDelay(3000);
        // VIDEO_CALL SECTION
```

```
        ImageIcon I7 = new
ImageIcon(ClassLoader.getSystemResource("chatting/application/icons/video.png")
);
        Image I8 = I7.getImage().getScaledInstance(30, 40,
Image.SCALE_DEFAULT);
        ImageIcon I9 = new ImageIcon(I8);
        JLabel L5 = new JLabel(I9);
        L5.setBounds(290, 15, 30, 40);
        P1.add(L5);

        // VOICE_CALL SECTION
        ImageIcon I10 = new
ImageIcon(ClassLoader.getSystemResource("chatting/application/icons/phone.png")
);
        Image I11 = I10.getImage().getScaledInstance(30, 40,
Image.SCALE_DEFAULT);
        ImageIcon I12 = new ImageIcon(I11);
        JLabel L6 = new JLabel(I12);
        L6.setBounds(350, 15, 30, 40);
        P1.add(L6);

        // SETTING SECTION IMAGE
        ImageIcon I13 = new
ImageIcon(ClassLoader.getSystemResource("chatting/application/icons/3icon.png")
);
        Image I14 = I13.getImage().getScaledInstance(10, 30,
Image.SCALE_DEFAULT);
        ImageIcon I15 = new ImageIcon(I14);
        JLabel L7 = new JLabel(I15);
        L7.setBounds(410, 20, 10, 30);
        P1.add(L7);

        // making SCROLLABLE PANEL to read messages
        A1 = new JPanel();
        A1.setBackground(new Color(220, 248, 198));
        A1.setFont(new Font("SAN_SERIF", Font.PLAIN, 17));
        // Adding Scrollable feature
        JScrollPane JS=new JScrollPane(A1);
        JS.setBounds(5, 75, 440, 560);
        JS.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED)
;
        JS.setBorder(BorderFactory.createEmptyBorder());
        F1.add(JS);

        // MESSAGE WRITING AREA
        T1 = new JTextField();
        T1.setBounds(10, 645, 355, 40);
        T1.setFont(new Font("SAN_SERIF", Font.PLAIN, 15));
        F1.add(T1);

        //Typing Status
        // If the keys are pressed while typing then change status to typing
        // KEY LISTENER FUNCTIONALITY
        T1.addKeyListener(new KeyAdapter() {
```

```
            public void keyPressed(KeyEvent ke) {
                L4.setText("Typing......");
                t.stop();
                typing = true;
            }

            public void keyReleased(KeyEvent ke) {
                L4.setText("Active Now");
                typing = false;
                if (!t.isRunning()) {
                    t.start();
                }

            }
        });
        // SEND BUTTON
        B1 = new JButton("Send");
        B1.setBounds(365, 645, 80, 40);
        B1.setBackground(new Color(18, 140, 126));
        B1.setForeground(new Color(220, 248, 198));
        B1.setFont(new Font("SAN_SERIF", Font.BOLD, 15));
        // adding functionality to button SEND
        B1.addActionListener(this);
        F1.add(B1);

        // since we will be designing our own layout
        // default layout provided is BORDER-LAYOUT
        F1.setLayout(null);
        F1.setUndecorated(true);
        F1.setSize(450, 700);
        F1.setLocation(300, 100);
        // GIVING BORDER TO FRAME
        F1.getRootPane().setBorder(
        BorderFactory.createMatteBorder(7,7,7,6, new Color(67, 90, 100))
);
        F1.setVisible(true);

    }

    // WHAT ACTIONS WILL BE PERFORMED WHEN SEND BUTTON IS PRESSED
    // Message transfer from one user to other
    // append that message in area panel A1
    public void actionPerformed(ActionEvent ae) {
        //overriding ACTION_LISTENER package function
        try {

            // Take Text from TEXTAREA and send it to other user
            String out = T1.getText();
            SendTextToFile(out,"Virat Kohli");
            dout.writeUTF(out);

            // make that message to append in AREA PANEL A1
            JPanel p = formatLabel(out, (int) 0);
```

```
            // The panel p contains message sent along with its TIME on the
right side
            A1.setLayout(new BorderLayout());
            JPanel right = new JPanel(new BorderLayout());
            right.setBackground(new Color(220, 248, 198));
            right.add(p, BorderLayout.LINE_END);
            vertical.add(right);
            A1.add(vertical, BorderLayout.PAGE_START);
            F1.validate();

            T1.setText("");
        } catch (Exception e) {
            System.out.println(e);
        }
    }

    // NOTE TO HAVE A DATABASE WE MUST MAKE A FILE
    // ADDING CHAT DATA TO THE CHAT FILE FOR BACK_UP PURPOSE
     public void SendTextToFile(String message,String name) throws
FileNotFoundException
     {
         try(PrintWriter pw = new PrintWriter(new FileOutputStream(new
File("Chat.txt"), true)); )
         {
             pw.println(name+" :- "+message);
         }
         catch(Exception e)
         {
             e.printStackTrace();
         }
     }
    // FOR SPEECH BUBBLES
    // Styling the message box area
    public static JPanel formatLabel(String out, int i) {
        JPanel k = new JPanel();
        k.setLayout(new BoxLayout(k, BoxLayout.Y_AXIS));
        // label created to display text
        // SETTING THE MAXIMUM WIDTH OF THE MESSAGE BOX
        JLabel L = new JLabel("<html><p style = \"width: 150px\">" + out +
"</p></html>");

        if (i == 1)
        {
            // that means we are sending message
            L.setBackground(new Color(52, 183, 241));
        } else
        {
            // we are reading the message
            L.setBackground(new Color(236, 229, 221));
        }
        L.setOpaque(true);
        L.setFont(new Font("Helvetica", Font.PLAIN, 17));
        L.setBorder(new EmptyBorder(10, 10, 10, 40));
        k.add(L);
```

```
        // label created to add time with the message
        // TIME WILL BE DISPLAYED ALONG WITH THE MESSAGE SENT
        Calendar cal = Calendar.getInstance();
        SimpleDateFormat sdf = new SimpleDateFormat("HH::mm");
        JLabel L2 = new JLabel();
        L2.setBackground(new Color(220, 248, 198));
        L2.setText(sdf.format(cal.getTime()));

        k.add(L2);
        return k;
    }

    // main method
    public static void main(String[] args) {
        new Server().F1.setVisible(true);

        // SOCKET PROGRAMMING or  NETWORK PROGRAMMING  PART
        String MessageInput = "";
        try {
            skt = new ServerSocket(6000);

            while (true) {
                // waiting for the connection to establish
                s = skt.accept();
                // connection established
                din = new DataInputStream(s.getInputStream());
                dout = new DataOutputStream(s.getOutputStream());

                while (true) {
                    MessageInput = din.readUTF();
                    // showing the input message
                     A1.setLayout(new BorderLayout());
                    JPanel p2 = formatLabel(MessageInput, (int) 1);
                    JPanel left = new JPanel(new BorderLayout());
                    left.setBackground(new Color(220, 248, 198));
                    left.add(p2, BorderLayout.LINE_START);
                    vertical.add(left);

                    A1.add(vertical,BorderLayout.PAGE_START);
                    F1.validate();
                }
            }
        } catch (Exception e) {
            System.out.println(e);
        }

    }
}
```

The Client Code :

The Code of the Client class is almost the same as Server class especially the GUI part. There are changes in socket programming part and hence the given code will represents the socket programming part of the CLIENT CLASS.

```
// SOCKET PROGRAMMING or NETWORK PROGRAMMING part
        try {
            String MessageInput = "";
            s = new Socket("127.0.0.1", 6000);
            din = new DataInputStream(s.getInputStream());
            dout = new DataOutputStream(s.getOutputStream());

            while (true) {
                // main method
                MessageInput = din.readUTF();
                A1.setLayout(new BorderLayout());
                JPanel p2 = formatLabel(MessageInput, (int) 1);
                JPanel left = new JPanel(new BorderLayout());
                left.setBackground(new Color(220, 248, 198));
                left.add(p2, BorderLayout.LINE_START);
                vertical.add(left);
                A1.add(vertical,BorderLayout.PAGE_START);
                F1.validate();
            }

        } catch (Exception e) {
            System.out.println(e);
        }
```

Conclusion

Thus, with all the functions and concepts working flawlessly one can definitely say that Desktop based Chatting Application is a wonderful Flawless project which on further modification and upgradation will become a complete application and can be used in for text chatting purpose. This mini Java desktop application uses the real world concepts of object oriented methodology like data abstract, data encapsulation, inheritance, socket programming, file handling, exception handling and Graphical User Interface (GUI) and hence it is a perfect example to depict the use of real world application made by using Objects Oriented Methodology.