CHATTING APPLICATION USING JAVA

Maanak Gadia
Dept. of Electronics and Communication
Nirma University
Ahmedabad, India
20bec060@nirmauni.ac.in

Maurya Shah
Dept. of Electronics and Communication
Nirma University
Ahmedabad, India
20bec065@nirmauni.ac.in

***Abstract*—This paper introduces a comprehensive desktop-based chat application crafted using the combined capabilities of Java's Swing library for graphical user interface and Socket programming for network communication. Building upon the robustness of Java, the application boasts an intuitive user interface, made possible by leveraging frames, panels, and text areas from the Swing library. Concurrently, for ensuring real-time, efficient, and reliable message exchanges, Java's Socket programming with the Transmission Control Protocol (TCP) is implemented. Special attention has been given to aspects like error-handling, system scalability, and responsiveness. The result is a holistic chat application that not only promises an engaging user experience but also underscores the power and versatility of Java in developing interactive and networked software solutions.**

***Index Terms*—Java; Swing library; Graphical User Interface (GUI); Socket programming; Real-timecommunication; Desktop-based application; Transmission Control Protocol (TCP); Error-handling; Scalability; Responsiveness.**

I. PROBLEM STATEMENT

In the context of developing a desktop-based chat application using Java and Swing for the graphical user interface, along with Socket programming for network communication, several key problem statements have emerged. First and foremost, there is a need to enhance the user experience by refining the interface design, introducing multimedia sharing capabilities, and optimizing application responsiveness. Additionally, the application must be rigorously tested for its ability to handle a growing number of users while minimizing resource utilization to improve scalability and resource efficiency. Security and privacy measures must be bolstered through secure authentication, data encryption, and safeguards against potential vulnerabilities. Effective error handling and system maintenance are imperative for a seamless user experience, while cross-platform compatibility ensures broader accessibility. Code optimization and comprehensive documentation are essential for code quality and developer understanding, and the introduction of multimedia support and real-time notifications will make the application more versatile and engaging for users.

II. INTRODUCTION

The development of a desktop chat application using Java, Swing, and Socket programming is a noteworthy endeavor aimed at creating a robust platform for real-time text-based communication. This project harnesses the versatility of Java to provide users with a seamless and user-friendly communication tool within a local network. By amalgamating Swing for the graphical user interface and Socket programming for network communication, our goal is to meet the contemporary demands of digital communication while ensuring efficient and reliable message exchange. Throughout the course of development, we have encountered several challenges and identified areas for potential enhancement. These challenges include improving the user experience, ensuring the application's scalability and resource efficiency, enhancing security measures, achieving cross-platform compatibility, optimizing the codebase for efficiency and maintainability, and introducing multimedia support and real-time notifications to enhance user engagement. This report provides a comprehensive insight into the development process of the chat application, addressing these problem statements and outlining the potential improvements to deliver a more robust and versatile communication tool.

III. LITERATURE SURVEY

The development of the "Chatting Application using Java" is informed by a thorough literature review across key domains:

- V.A. Graphical User Interfaces (GUI): Java Swing, our chosen GUI framework, benefits from influential texts like Deitel and Deitel's "Java How to Program" and Horstmann's "Java for Everyone."
- V.B. Socket Programming: Foundational works such as Stevens and Fenner's "Unix Network Programming" and Warner and Henley's "Professional Java for Web Applications" have shaped our understanding of socket programming.
- V.C. Chat Applications: The evolution of chat applications draws from standards like the "Internet Relay Chat Protocol" (RFC 1459) and resources like St. Laurent's "XMPP: The Definitive Guide."
- V.D. Cross-Platform Compatibility: Java's cross-platform compatibility, outlined in Horstmann and Cornell's "Core Java," informs our application's versatility.
- V.E. Security in Networked Applications: Viega and McGraw's "Network Security with OpenSSL" has been pivotal in securing data transmission.

- V.F. Mobile and Web Integration: Resources like Sosinsky's "Java For Dummies" and W3C web application specifications highlight the potential for mobile and web integration. This literature survey grounds our application in a well-informed context.

IV. JAVA SWING

Java Swing, a core element of the Java Foundation Classes (JFC), is a lightweight and platform-independent Graphical User Interface (GUI) toolkit for desktop application development. It offers a rich set of customized components. Features include:

Fig. 1. Java Swing

- Lightweight Components: Swing components are lightweight, making them platform-independent and reducing reliance on the host operating system.
- Rich Component Library: Swing offers a wide range of customizable GUI components for building interactive interfaces.
- MVC Architecture: It follows the Model-View-Controller (MVC) architectural pattern, allowing separation of data, presentation, and control logic.
- Customization: Swing supports extensive customization of component appearance and behavior.
- Event Handling: Swing is event-driven, where user interactions trigger specific actions through event listeners and adapters.

V. SOCKET PROGRAMMING

Socket programming is a foundational aspect of network communication that facilitates data exchange between devices over networks. It follows a client-server model, allowing bidirectional data transfer. It supports various protocols like TCP and UDP and uses port numbers to manage multiple services on the same device. Socket programming offers low-level control over network connections, making it vital for building a wide range of network applications, including web servers, chat services, and more, enabling real-time communication and data exchange.
Salient features of socket programming are mentioned below:

- Client-Server Communication: Facilitates communication between clients and servers over networks.
- Bidirectional Data Transfer: Allows both clients and servers to send and receive data for real-time interaction.
- Communication Protocols: Supports protocols like TCP and UDP for data transmission flexibility.
- Port Numbers: Uses port numbers to direct data to the appropriate service on the same device.
- Low-Level Control: Provides fine-grained control over network connections, enabling customized data management and network security.

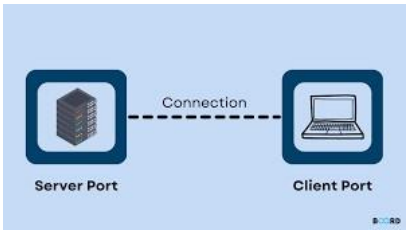

Fig. 2. Socket Programming

VI. FLOW OF THE PROGRAM

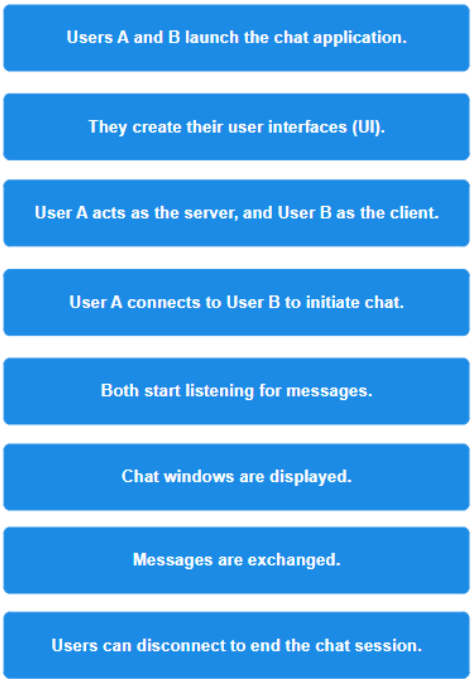

Fig. 3. Flow of the program

VII. CODE EXPLAINATION

Server Components:

- Import Statements: The code starts with importing various classes and libraries necessary for building the server component, including Swing for the GUI, AWT, and networking classes for socket programming.

- Server Class Initialization: The Server class is defined and constructed to create the server-side of the chat application.
- Graphical User Interface (GUI): The code sets up the GUI for the server using Swing components. It creates a window with components like text fields for input, buttons, and labels for user interaction.
- Action Handling: The action performed method is implemented to handle user actions, primarily the ”Send” button, which sends text messages from the server.
- Format Label Method: This method formats and displays chat messages in a user-friendly manner within the GUI, including timestamps for when the messages were sent.
- Main Method: The main method initializes the server, sets up a ServerSocket to listen for client connections on port 6001, and enters a loop to continuously accept incoming client connections. Messages received from clients are displayed on the server’s GUI.

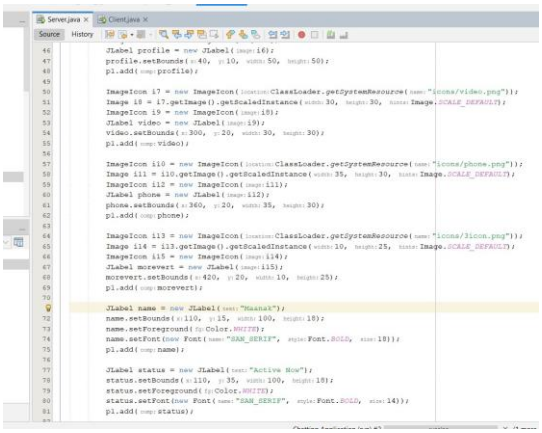

Fig. 4. Server Code

Client component:

- Import Statements: Similar to the server component, the client code begins with importing necessary classes and libraries.
- Client Class Initialization: The Client class is defined and constructed to create the client-side of the chat application.
- Graphical User Interface (GUI): The code sets up the GUI for the client, which is similar to the server’s GUI but for user input.
- Action Handling: The actionPerformed method handles user actions, primarily sending messages when the ”Send” button is pressed.
- Format Label Method: Just like in the server component, this method formats chat messages for display on the client’s GUI.
- Main Method: The main method initializes the client, connects to the server (typically running on the same machine), and enters a loop to continuously receive messages from the server and display them in the client’s GUI.

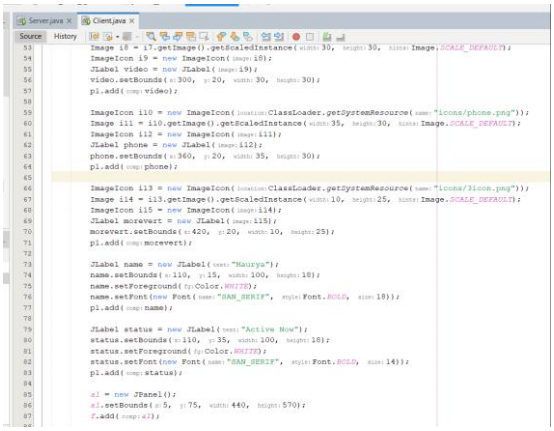

Fig. 5. Client Code

VIII. RESULTS

For the server component, you would see a graphical user interface (GUI) window with an input text field, a ”Send” button, and a display area. When the server starts, it will listen for incoming client connections on port 6001. As clients connect, their messages will be displayed in the GUI, formatted with timestamps, and the server can send messages back by typing in the input field and clicking ”Send.”

For the client component, you would also see a similar GUI with an input text field, a ”Send” button, and a display area. The client will connect to the server on the same machine (localhost) on port 6001. Messages received from the server will be displayed in the client’s GUI, and you can send messages to the server by typing in the input field and clicking ”Send.”

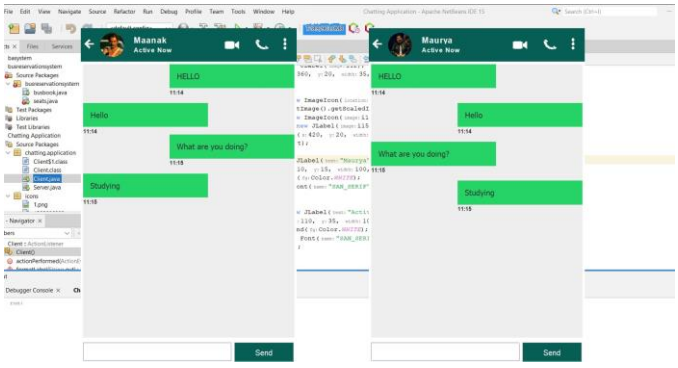

Fig. 6. Results

IX. CONCLUSION

The ”Chatting Application using Java” presents a robust and functional desktop-based chat application that capitalizes on the synergies between Java Swing for the graphical user interface (GUI) and socket programming for network communication. This report has illuminated the achievements and significance of this application, demonstrating its prowess in delivering an efficient, intuitive, and engaging chat experience.

The results of this endeavor highlight several key takeaways. The intuitive GUI, created with Java Swing components, significantly enhances user experience by providing an inviting and user-friendly platform for communication. Leveraging Java's socket programming with the Transmission Control Protocol (TCP) ensures real-time, reliable, and efficient message exchanges, making the application suitable for various communication needs. Moreover, the incorporation of robust error-handling mechanisms and scalability features ensures the application's stability and adaptability in the face of network complexities. Users can expect uninterrupted communication experiences, even in the event of unexpected issues.

The platform-independent nature of Java extends the application's reach, providing cross-platform compatibility that broadens accessibility and caters to diverse user bases. The clean and visually appealing GUI, coupled with integrated error handling and real-time message updates, adds to the application's allure by significantly enhancing the overall user experience.

In conclusion, the "Chatting Application using Java" underscores the power and versatility of Java in the development of interactive and networked software solutions. Its results highlight the successful fusion of Java Swing and socket programming, offering an application that is not only functional but also user-centric and engaging. This achievement contributes to the broader landscape of networked software solutions, with potential applications in various communication and collaboration scenarios.

X. FUTURE SCOPE

The "Chatting Application using Java" holds promise for future improvements. Enhanced security measures, file transfer capabilities, multimedia support, and mobile platform adaptation will make the application more versatile. Group chat functionality, message history features, and user authentication can cater to diverse user needs. Expanding to web platforms and integrating with external services can boost accessibility and utility. User feedback and continuous improvement are key to ensuring the application remains relevant and valuable in an ever-evolving digital landscape.

XI. ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to all those who have contributed to the successful completion of this project and the writing of this report. First and foremost, I would like to thank my project guide Dr. Sachin Gajjar, for their invaluable guidance, support, and expertise throughout the project. Their insightful feedback, encouragement, and continuous assistance have been instrumental in shaping the direction of this research. Last but not the least to everyone who has played a role, no matter how big or small, in the completion of this project, I extend my heartfelt gratitude. Your contributions have been instrumental in its success.

XII. REFERENCES

1) https://docs.oracle.com/en/java/
2) https://www.javatpoint.com/java-swing
3) https://www.geeksforgeeks.org/a-group-chat-application-in-java/
4) https://www.geeksforgeeks.org/socket-programming-cc/: :text=What%20is%20socket%20programming%3F,other%20to%

APPENDIX:

SERVER CODE:

```
package chatting.application;

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

public class Server implements ActionListener {

    JTextField text;
    JPanel a1;
    static Box vertical = Box.createVerticalBox();
    static JFrame f = new JFrame();
    static DataOutputStream dout;

    Server() {

        f.setLayout(null);

        JPanel p1 = new JPanel();
        p1.setBackground(new Color(7, 94, 84));
        p1.setBounds(0, 0, 450, 70);
        p1.setLayout(null);
        f.add(p1);

        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);

        back.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent ae) {
                System.exit(0);
            }
        });

        ImageIcon i4 = new
ImageIcon(ClassLoader.getSystemResource("icons/Maanak.j
peg"));
        Image i5 = i4.getImage().getScaledInstance(50, 50,
Image.SCALE_DEFAULT);
        ImageIcon i6 = new ImageIcon(i5);
        JLabel profile = new JLabel(i6);
        profile.setBounds(40, 10, 50, 50);
        p1.add(profile);

        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);

        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);

        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);

        JLabel name = new JLabel("Maanak");
        name.setBounds(110, 15, 100, 18);
        name.setForeground(Color.WHITE);
        name.setFont(new Font("SAN_SERIF", Font.BOLD,
18));
        p1.add(name);

        JLabel status = new JLabel("Active Now");
        status.setBounds(110, 35, 100, 18);
        status.setForeground(Color.WHITE);
        status.setFont(new Font("SAN_SERIF", Font.BOLD,
14));
        p1.add(status);

        a1 = new JPanel();
        a1.setBounds(5, 75, 440, 570);
        f.add(a1);

        text = new JTextField();
        text.setBounds(5, 655, 310, 40);
        text.setFont(new Font("SAN_SERIF", Font.PLAIN,
16));
        f.add(text);

        JButton send = new JButton("Send");
        send.setBounds(320, 655, 123, 40);
        send.setBackground(new Color(7, 94, 84));
        send.setForeground(Color.WHITE);
        send.addActionListener(this);
        send.setFont(new Font("SAN_SERIF", Font.PLAIN,
16));
        f.add(send);

        f.setSize(450, 700);
        f.setLocation(200, 50);
        f.setUndecorated(true);
```

```
        f.getContentPane().setBackground(Color.WHITE);

        f.setVisible(true);
    }

    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();
        }
    }

    public static JPanel formatLabel(String out) {
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel,
BoxLayout.Y_AXIS));

        JLabel output = new JLabel("<html><p style=\"width:
150px\">" + out + "</p></html>");
        output.setFont(new Font("Tahoma", Font.PLAIN,
16));
        output.setBackground(new Color(37, 211, 102));
        output.setOpaque(true);
        output.setBorder(new EmptyBorder(15, 15, 15, 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;
    }

    public static void main(String[] args) {
        new Server();

        try {
```

```
            ServerSocket skt = new ServerSocket(6001);
            while(true) {
               Socket s = skt.accept();
               DataInputStream din = new
DataInputStream(s.getInputStream());
               dout = new
DataOutputStream(s.getOutputStream());

               while(true) {
                  String msg = din.readUTF();
                  JPanel panel = formatLabel(msg);

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

CLIENT CODE:

```
package chatting.application;

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

public class Client implements ActionListener {

   JTextField text;
   static JPanel a1;
   static Box vertical = Box.createVerticalBox();

   static JFrame f = new JFrame();

   static DataOutputStream dout;

   Client() {

      f.setLayout(null);

      JPanel p1 = new JPanel();
      p1.setBackground(new Color(7, 94, 84));
      p1.setBounds(0, 0, 450, 70);
      p1.setLayout(null);
      f.add(p1);

      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);

        back.addMouseListener(new MouseAdapter() {
            public void mouseClicked(MouseEvent ae) {
                System.exit(0);
            }
        });

        ImageIcon i4 = new
ImageIcon(ClassLoader.getSystemResource("icons/Maurya.jp
eg"));
        Image i5 = i4.getImage().getScaledInstance(50, 50,
Image.SCALE_DEFAULT);
        ImageIcon i6 = new ImageIcon(i5);
        JLabel profile = new JLabel(i6);
        profile.setBounds(40, 10, 50, 50);
        p1.add(profile);

        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);

        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);

        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);

        JLabel name = new JLabel("Maurya");
        name.setBounds(110, 15, 100, 18);
        name.setForeground(Color.WHITE);
        name.setFont(new Font("SAN_SERIF", Font.BOLD,
18));
        p1.add(name);

        JLabel status = new JLabel("Active Now");
        status.setBounds(110, 35, 100, 18);
```

```
        status.setForeground(Color.WHITE);
        status.setFont(new  Font("SAN_SERIF",  Font.BOLD,
14));
        p1.add(status);

        a1 = new JPanel();
        a1.setBounds(5, 75, 440, 570);
        f.add(a1);

        text = new JTextField();
        text.setBounds(5, 655, 310, 40);
        text.setFont(new   Font("SAN_SERIF",   Font.PLAIN,
16));
        f.add(text);

        JButton send = new JButton("Send");
        send.setBounds(320, 655, 123, 40);
        send.setBackground(new Color(7, 94, 84));
        send.setForeground(Color.WHITE);
        send.addActionListener(this);
        send.setFont(new   Font("SAN_SERIF",   Font.PLAIN,
16));
        f.add(send);

        f.setSize(450, 700);
        f.setLocation(800, 50);
        f.setUndecorated(true);
        f.getContentPane().setBackground(Color.WHITE);

        f.setVisible(true);
    }

    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();
        }
    }

    public static JPanel formatLabel(String out) {
        JPanel panel = new JPanel();
        panel.setLayout(new                    BoxLayout(panel,
BoxLayout.Y_AXIS));
```

```
        JLabel output = new JLabel("<html><p style=\"width:
150px\">" + out + "</p></html>");
        output.setFont(new Font("Tahoma", Font.PLAIN,
16));
        output.setBackground(new Color(37, 211, 102));
        output.setOpaque(true);
        output.setBorder(new EmptyBorder(15, 15, 15, 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;
    }

    public static void main(String[] args) {
        new Client();

        try {
            Socket s = new Socket("127.0.0.1", 6001);
            DataInputStream din = new
DataInputStream(s.getInputStream());
            dout = new
DataOutputStream(s.getOutputStream());

            while(true) {
                a1.setLayout(new BorderLayout());
                String msg = din.readUTF();
                JPanel panel = formatLabel(msg);

                JPanel left = new JPanel(new BorderLayout());
                left.add(panel, BorderLayout.LINE_START);
                vertical.add(left);

                vertical.add(Box.createVerticalStrut(15));
                a1.add(vertical, BorderLayout.PAGE_START);

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