

JAVA PROGRAMMING

[ IT201 ]

HOME ASSIGNMENT(HA/S/V/Q)

**NAME:- JAYENDRA**

**E.NO:- A35705222012**

**COURSE:- B.TECH (CSE) SECTION “A”**

**BATCH:- SEMESTER 4 (2022-2026)**

**FACULTY:- DR. PALLAB BANERJEE**

**FACULTY SIGNATURE**

**PROJECT OVERVIEW**

**Brief Introduction to the Project**

This project revolves around the development of a dynamic and user-friendly chat application using the Java programming language. In today's fast-paced digital era, effective communication is paramount, and our application aims to facilitate seamless interaction among users in a group setting.

**Key Features:**

1.) Real-time Messaging: Experience instantaneous communication with friends, colleagues, or team members, fostering efficient collaboration regardless of geographical barriers.

2.) Intuitive GUI: Our user-friendly graphical interface ensures ease of navigation and interaction, making it accessible to users of all skill levels.

3.) Scalability: Designed to scale effortlessly over LAN connected machines by leveraging their IP addresses, JavaNetChat adapts to your growing communication needs with ease.

**Overview of the Problem Statement**

Traditional communication methods often lack the immediacy and convenience required for modern-day collaboration. Recognizing this gap, we set out to create a solution that enables real-time messaging while prioritizing simplicity and accessibility.

**Importance of Developing a Chat Application in JAVA**

Java, renowned for its versatility and robustness, serves as the foundation for our project. Leveraging Java's rich ecosystem of libraries and frameworks, we aim to deliver a reliable and scalable chat application that meets the diverse needs of our users. Moreover, Java's platform independence ensures compatibility across various devices and operating systems, enhancing the accessibility of our solution.

**Technologies Used**

* Java: Main programming language used for development
* Socket Programming: Used for communication between client and server
* Swing: Java GUI toolkit used for creating the graphical user interface
* Networking: Utilized for establishing connections over LAN

**CODE**

**SOCKET CLIENT**

import javax.swing.\***;**import java.awt.\***;**import java.awt.event.ActionEvent**;**import java.awt.event.ActionListener**;**import java.io.BufferedReader**;**import java.io.InputStreamReader**;**import java.io.PrintWriter**;**import java.net.Socket**;**@SuppressWarnings("serial")  
public class SocketClient extends JFrame implements ActionListener**,** Runnable {  
 JTextArea textArea = new JTextArea()**;** JScrollPane jp = new JScrollPane(textArea)**;** JTextField input\_Text = new JTextField()**;** JMenuBar menuBar = new JMenuBar()**;** Socket sk**;** BufferedReader br**;** PrintWriter pw**;** public SocketClient() {  
 super("DialogueDuo")**;** setFont(new Font("Arial Black"**,** Font.*PLAIN***, 12**))**;** setForeground(new Color(**0, 0, 51**))**;** setBackground(new Color(**51, 0, 0**))**;** textArea.setToolTipText("Chat History")**;** textArea.setForeground(new Color(**50, 205, 50**))**;** textArea.setEditable(false)**;** textArea.setFont(new Font("Monospaced"**,** Font.*BOLD***, 13**))**;** textArea.setBackground(new Color(**0, 0, 0**))**;** getContentPane().add(jp**,** "Center")**;** input\_Text.setText("Enter your Message:")**;** input\_Text.setToolTipText("Enter your Message")**;** input\_Text.setForeground(new Color(**0, 0, 0**))**;** input\_Text.setFont(new Font("Tahoma"**,** Font.*BOLD***, 11**))**;** input\_Text.setBackground(new Color(**230, 230, 250**))**;** getContentPane().add(input\_Text**,** "South")**;** setSize(**325, 411**)**;** setVisible(true)**;** input\_Text.requestFocus()**;** //Place cursor at run time, work after screen is shown  
  
 setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*)**;** input\_Text.addActionListener(this)**;** //Event registration  
 }  
  
 public void serverConnection() {  
 try {  
 String IP = JOptionPane.*showInputDialog*(this**,** "Please enter a server IP."**,** JOptionPane.*INFORMATION\_MESSAGE*)**;** sk = new Socket(IP**, 1234**)**;** String name = JOptionPane.*showInputDialog*(this**,** "Please enter a nickname"**,** JOptionPane.*INFORMATION\_MESSAGE*)**;** //read  
 br = new BufferedReader(new InputStreamReader(sk.getInputStream()))**;** //writing  
 pw = new PrintWriter(sk.getOutputStream()**,** true)**;** pw.println(name)**;** // Send to server side  
  
 new Thread(this).start()**;** } catch (Exception e) {  
 System.*out*.println(e + " Socket Connection error")**;** }  
 }  
  
 public static void main(String[] args) {  
 new SocketClient().serverConnection()**;** //Method call at the same time object creation  
 }  
  
 @Override  
 public void run() {  
 String data = null**;** try {  
 while ((data = br.readLine()) != null) {  
 textArea.append(data + "\n")**;** //textArea Decrease the position of the box's scroll bar by the length of the text entered  
 textArea.setCaretPosition(textArea.getText().length())**;** }  
 } catch (Exception e) {  
 System.*out*.println(e + "--> Client run fail")**;** }  
 }  
  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String data = input\_Text.getText()**;** pw.println(data)**;** // Send to server side  
 input\_Text.setText("")**;** }

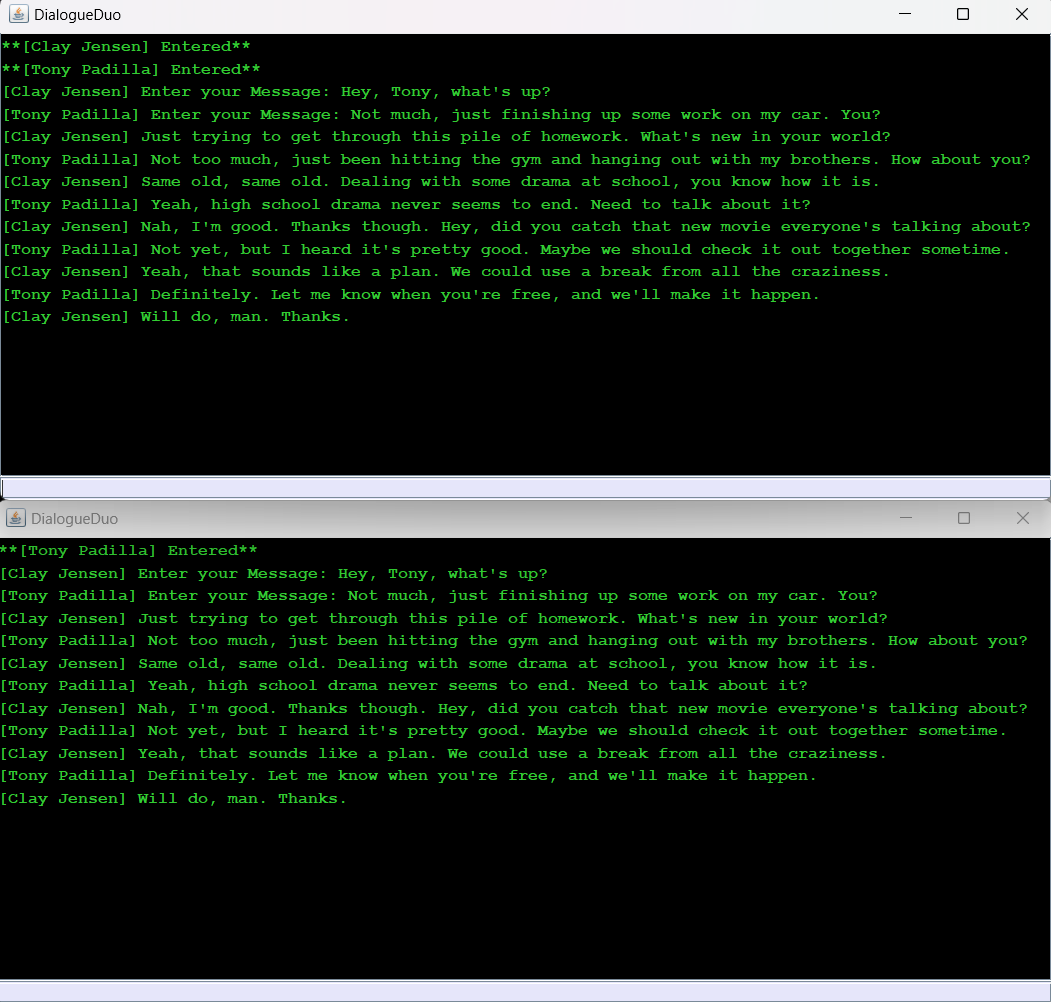
**SOCKET SERVER**

import java.io.BufferedReader**;**import java.io.IOException**;**import java.io.InputStreamReader**;**import java.io.PrintWriter**;**import java.net.InetAddress**;**import java.net.ServerSocket**;**import java.net.Socket**;**import java.util.ArrayList**;**public class SocketServer {  
 ServerSocket server**;** Socket sk**;** InetAddress addr**;** ArrayList<ServerThread> list = new ArrayList<ServerThread>()**;** public SocketServer() {  
 try {  
 addr = InetAddress.*getByName*("127.0.0.1")**;** //addr = InetAddress.getByName("192.168.43.1");  
   
 server = new ServerSocket(**1234,50,**addr)**;** System.*out*.println("\n Waiting for Client connection")**;** SocketClient.*main*(null)**;** while(true) {  
 sk = server.accept()**;** System.*out*.println(sk.getInetAddress() + " connect")**;** //Thread connected clients to ArrayList  
 ServerThread st = new ServerThread(this)**;** addThread(st)**;** st.start()**;** }  
 } catch(IOException e) {  
 System.*out*.println(e + "-> ServerSocket failed")**;** }  
 }  
  
 public void addThread(ServerThread st) {  
 list.add(st)**;** }  
  
 public void removeThread(ServerThread st){  
 list.remove(st)**;** //remove  
 }  
  
 public void broadCast(String message){  
 for(ServerThread st : list){  
 st.pw.println(message)**;** }  
 }  
  
 public static void main(String[] args) {  
 new SocketServer()**;** }  
}  
  
class ServerThread extends Thread {  
 SocketServer server**;** PrintWriter pw**;** String name**;** public ServerThread(SocketServer server) {  
 this.server = server**;** }  
  
 @Override  
 public void run() {  
 try {  
 // read  
 BufferedReader br = new BufferedReader(new InputStreamReader(server.sk.getInputStream()))**;** // writing  
 pw = new PrintWriter(server.sk.getOutputStream()**,** true)**;** name = br.readLine()**;** server.broadCast("\*\*["+name+"] Entered\*\*")**;** String data**;** while((data = br.readLine()) != null ){  
 if(data == "/list"){  
 pw.println("a")**;** }  
 server.broadCast("["+name+"] "+ data)**;** }  
 } catch (Exception e) {  
 //Remove the current thread from the ArrayList.  
 server.removeThread(this)**;** server.broadCast("\*\*["+name+"] Left\*\*")**;** System.*out*.println(server.sk.getInetAddress()+" - ["+name+"] Exit")**;** System.*out*.println(e + "---->")**;** }  
 }  
}

**XML FILE**

<?xml version="1.0" encoding="UTF-8"?>  
<module type="JAVA\_MODULE" version="4">  
 <component name="NewModuleRootManager" inherit-compiler-output="true">  
 <exclude-output />  
 <content url="file://$MODULE\_DIR$">  
 <sourceFolder url="file://$MODULE\_DIR$" isTestSource="false" />  
 </content>  
 <orderEntry type="inheritedJdk" />  
 <orderEntry type="sourceFolder" forTests="false" />  
 </component>  
</module>

**OUTPUT**



**FUTURE ENHANCEMENTS**

Our Chat Application using Java Socket Programming has laid a solid foundation for seamless communication. However, envisioning its future, we see boundless opportunities for enhancement and innovation. Here's a glimpse into what lies ahead:

1. Enhanced Security: Implementing end-to-end encryption to ensure secure messaging, safeguarding privacy, and confidentiality in every exchange.

2. Advanced Collaboration: Introducing file sharing capabilities, enabling users to effortlessly share documents, images, and media, fostering richer collaboration experiences.

3. User Authentication: Strengthening the application with user authentication and authorization mechanisms, ensuring only authorized users gain access, enhancing control and security.

4. Customization Options: Offering personalized themes and customization features, allowing users to tailor their chat environment to reflect their style and preferences.

5. Integration with Multimedia: Integrating multimedia support for sharing videos, audio files, and more, enriching communication with diverse content formats.

**CONCLUSION**

In conclusion, our Chat Application using Java Socket Programming represents more than just a tool for messaging; it embodies the essence of connectivity and collaboration. With its simple yet robust architecture, intuitive user interface, and scalable design, the application serves as a gateway to seamless communication.

Throughout this journey, I have witnessed the power of Java in facilitating real-time messaging and the versatility of Socket Programming in enabling communication between client and server. By harnessing these technologies, I have created a platform that transcends geographical boundaries, allowing users to connect and collaborate effortlessly.

As I reflect on the significance of this project, I am reminded of the ever-expanding landscape of communication and the role that innovative solutions play in shaping it. My Chat Application stands as a testament to the endless possibilities that lie at the intersection of technology and human interaction.

I extend my gratitude to all those who have been a part of this journey – from the developers who brought the vision to life to the users who breathe life into the application with every message exchanged. Together, we have built more than just a chat application; we have built a community, united by the desire to connect, collaborate, and communicate.

As I look towards the future, I remain committed to pushing the boundaries of what is possible, continuously striving to enhance the user experience and empower connectivity in all its forms.

**Connect. Collaborate. Communicate.**