## A chat room application - (TCP SOCKET)

## MIS – 642303019

## ChatClient.java

import java.io.\*;

import java.net.\*;

public class ChatClient {

    private static final String SERVER\_ADDRESS = "localhost";

    private static final int SERVER\_PORT = 12345;

    public static void main(String[] args) {

        try {

            Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT);

            BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

            BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));

            new Thread(() -> {

                try {

                    String message;

                    while ((message = in.readLine()) != null) {

                        System.out.println(message);

                    }

                } catch (IOException e) {

                    System.err.println("Error reading from server: " + e.getMessage());

                }

            }).start();

            String message;

            while ((message = userInput.readLine()) != null) {

                out.println(message);

            }

        } catch (IOException e) {

            System.err.println("Error connecting to the server: " + e.getMessage());

        }

    }

}

## ChatServer.java

import java.io.\*;

import java.net.\*;

import java.text.SimpleDateFormat;

import java.util.\*;

public class ChatServer {

    private static Set<ClientHandler> clientHandlers = Collections.synchronizedSet(new HashSet<>());

    public static void main(String[] args) throws IOException {

        ServerSocket serverSocket = new ServerSocket(12345);

        System.out.println("Chat Server started...");

        while (true) {

            Socket socket = serverSocket.accept();

            ClientHandler clientHandler = new ClientHandler(socket);

            clientHandlers.add(clientHandler);

            new Thread(clientHandler).start();

        }

    }

    // Broadcast message to all clients

    public static void broadcastMessage(String message, ClientHandler excludeUser) {

        synchronized (clientHandlers) {

            for (ClientHandler clientHandler : clientHandlers) {

                if (clientHandler != excludeUser) {

                    clientHandler.sendMessage(message);

                }

            }

        }

    }

    // Remove a client from the chat

    public static void removeClient(ClientHandler clientHandler) {

        synchronized (clientHandlers) {

            clientHandlers.remove(clientHandler);

        }

    }

}

class ClientHandler implements Runnable {

    private Socket socket;

    private PrintWriter out;

    private BufferedReader in;

    private String userName;

    public ClientHandler(Socket socket) {

        this.socket = socket;

    }

    @Override

    public void run() {

        try {

            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            out = new PrintWriter(socket.getOutputStream(), true);

            // Get user name

            out.println("Enter your name: ");

            userName = in.readLine();

            System.out.println(userName + " has joined the chat.");

            ChatServer.broadcastMessage(userName + " has joined the chat.", this);

            String message;

            while ((message = in.readLine()) != null) {

                SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");

                String time = sdf.format(new Date());

                String fullMessage = "[" + time + "] " + userName + ": " + message;

                System.out.println(fullMessage); // Server-side log

                ChatServer.broadcastMessage(fullMessage, this); // Broadcast to others

            }

        } catch (IOException e) {

            System.err.println("Error in ClientHandler: " + e.getMessage());

        } finally {

            try {

                socket.close();

            } catch (IOException e) {

                System.err.println("Error closing socket: " + e.getMessage());

            }

            ChatServer.removeClient(this);

            System.out.println(userName + " has left the chat.");

            ChatServer.broadcastMessage(userName + " has left the chat.", this);

        }

    }

    // Send a message to the client

    public void sendMessage(String message) {

        out.println(message);

    }

}

## OUTPUT

## 