**Server.java**

import java.io.\*;

import java.net.\*;

import java.util.\*;

import org.json.JSONObject;

public class Server {

    private static final String SERVER\_IP = "10.70.33.162";

    private static final int SERVER\_PORT = 8000;

    private static final Map<Socket, String> clients = new HashMap<>();

    public static void main(String[] args) {

        System.out.println("Server started on " + SERVER\_IP + ":" + SERVER\_PORT);

        try (ServerSocket serverSocket = new ServerSocket(SERVER\_PORT, 50, InetAddress.getByName(SERVER\_IP))) {

            while (true) {

                Socket clientSocket = serverSocket.accept();

                System.out.println(

                        "New client connected: " + clientSocket.getInetAddress() + ":" + clientSocket.getPort());

                synchronized (clients) {

                    clients.put(clientSocket,

                            clientSocket.getInetAddress().getHostAddress() + ":" + clientSocket.getPort());

                    updateClientList();

                }

                new Thread(() -> handleClient(clientSocket)).start();

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    private static void handleClient(Socket clientSocket) {

        try (BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

                PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)) {

            String message;

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

                JSONObject data = new JSONObject(message);

                if ("message".equals(data.getString("type"))) {

                    String[] recipient = data.getString("recipient\_ip").split(":", 2);

                    String recipientIp = recipient[0].trim();

                    int recipientPort = Integer.parseInt(recipient[1].trim());

                    Socket recipientSocket = findClientSocket(recipientIp, recipientPort);

                    if (recipientSocket != null) {

                        PrintWriter recipientOut = new PrintWriter(recipientSocket.getOutputStream(), true);

                        JSONObject response = new JSONObject();

                        response.put("type", "message");

                        response.put("message", data.getString("message"));

                        response.put("sender\_ip", data.getString("sender\_ip"));

                        recipientOut.println(response.toString());

                    }

                } else if ("disconnect".equals(data.getString("type"))) {

                    synchronized (clients) {

                        clients.remove(clientSocket);

                        updateClientList();

                    }

                    break;

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        } finally {

            try {

                clientSocket.close();

            } catch (IOException e) {

                e.printStackTrace();

            }

        }

    }

    private static Socket findClientSocket(String ip, int port) {

        synchronized (clients) {

            for (Map.Entry<Socket, String> entry : clients.entrySet()) {

                String[] clientInfo = entry.getValue().split(":", 2);

                String clientIp = clientInfo[0].trim();

                int clientPort = Integer.parseInt(clientInfo[1].trim());

                if (ip.equals(clientIp) && port == clientPort) {

                    return entry.getKey();

                }

            }

        }

        return null;

    }

    private static void updateClientList() {

        List<String> clientList = new ArrayList<>();

        synchronized (clients) {

            for (String clientInfo : clients.values()) {

                clientList.add(clientInfo);

            }

            clientList.add(SERVER\_IP + ":" + SERVER\_PORT);

        }

        JSONObject update = new JSONObject();

        update.put("type", "client\_list");

        update.put("client\_list", clientList);

        for (Socket clientSocket : clients.keySet()) {

            try {

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

                out.println(update.toString());

            } catch (IOException e) {

                e.printStackTrace();

            }

        }

    }

}

**Client.java**

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

import org.json.JSONObject;

public class Client {

    private String host;

    private int port;

    private Socket clientSocket;

    private String clientIp;

    private int clientPort;

    public Client(String host, int port) throws IOException {

        this.host = host;

        this.port = port;

        this.clientSocket = new Socket(this.host, this.port);

        this.clientIp = clientSocket.getLocalAddress().getHostAddress();

        this.clientPort = clientSocket.getLocalPort();

    }

    public void sendMessage() {

        try (PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)) {

            Scanner scanner = new Scanner(System.in);

            while (true) {

                System.out.println(

                        "Enter message (format: ip:port:message or 'server:port:message' to send to server): ");

                String messageInput = scanner.nextLine();

                String[] parts = messageInput.split(":", 3);

                String recipientIp = parts[0].trim();

                int recipientPort = Integer.parseInt(parts[1].trim());

                String message = parts[2].trim();

                if (recipientIp.equalsIgnoreCase("server")) {

                    recipientIp = host;

                    recipientPort = port;

                }

                JSONObject data = new JSONObject();

                data.put("type", "message");

                data.put("message", message);

                data.put("sender\_ip", clientIp + ":" + clientPort);

                data.put("recipient\_ip", recipientIp + ":" + recipientPort);

                out.println(data.toString());

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    public void receiveMessage() {

        try (BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()))) {

            String message;

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

                JSONObject data = new JSONObject(message);

                if (data.getString("type").equals("message")) {

                    System.out.println("\nReceived message from " + data.getString("sender\_ip") + ": "

                            + data.getString("message"));

                } else if (data.getString("type").equals("client\_list")) {

                    System.out.println("\nClient list: " + data.getJSONArray("client\_list").toString() + "\n");

                }

                System.out

                        .print("Enter message (format: ip:port:message or 'server:port:message' to send to server): ");

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    public void run() {

        System.out.println("Your IP: " + clientIp + ":" + clientPort);

        new Thread(this::receiveMessage).start();

        new Thread(this::sendMessage).start();

    }

    public static void main(String[] args) {

        try {

            Client client = new Client("10.70.33.162", 8000);

            client.run();

        } catch (IOException e) {

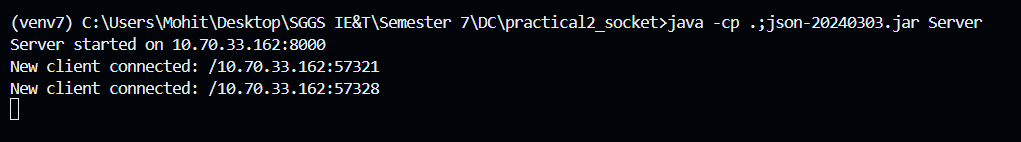
            e.printStackTrace();

        }

    }

}

**Output:**

A computer code on a black background

Description automatically generatedA computer screen shot of text

Description automatically generated