Source Code	Keterangan	Penjelasan

Client :

```java
import java.io.*;
import java.net.*;
import java.net.Socket;
import java.util.Scanner;

public class Client {
    public static void main(String[] args) {
        while (true) {
            try {
                // Mencoba membuat koneksi ke server dengan alamat "localhost" dan port 12345
                Socket socket = new Socket("localhost", 12345);
                System.out.println("Terhubung ke server. ");

                // Membuat aliran untuk menerima pesan dari server
                BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

                // Membuat aliran untuk mengirim pesan ke server
                PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

                // Membaca pesan dari pengguna
                Scanner scanner = new Scanner(System.in);
                System.out.print("Masukkan nama anda : ");

                // Thread untuk menerima pesan dari server
                Thread receiveThread = new Thread(() -> {
                    try {
                        String message;
                        while ((message = in.readLine()) != null) {

                            // Menampilkan pesan dari server
                            System.out.println("" + message);
                        }
                    } catch (IOException e) {

                        // Menampilkan pesan jika koneksi dengan server terputus
                        System.out.println("Server : Disconnect");
                        System.out.println(e.getMessage());

                    }
                });
                receiveThread.start();

                while (true) {
                    System.out.print("");

                    // Membaca pesan dari pengguna dan mengirimkannya ke server
                    String message = scanner.nextLine();
                    out.println(message);
                }
            } catch (IOException e) {
                // Menampilkan pesan jika gagal terhubung ke server
                System.out.println("Gagal terhubung ke server. Akan mencoba lagi dalam beberapa detik.");
                try {
                    Thread.sleep(5000); // Tunggu 5 detik sebelum mencoba menghubungkan ulang
                } catch (InterruptedException ex) {
                    ex.printStackTrace();
                }
            }
        }
    }
}
```

Client :

Inisialisasi dan Pembuatan Socket untuk koneksi ke server

- Socket dibuat pada program client dan server untuk memulai komunikasi
- Socket client akan terhubung ke alamat IP dan nomor port yang ditentukan untuk server.

Aliran Pengiriman & Penerimaan pesan

- Setelah koneksi terbentuk, client & server dapat transfer pesan melalui socket

Membaca Inputan Pesan berupa inisial "nama client" saat setelah terhubung ke server

Thread penerimaan & penampilan pesan

Penanganan Kesalahan (Error Handling)

- Socket Programming menggunakan Error Handling untuk mengatasi kondisi yang tidak diinginkan, seperti koneksi terputus/kegagalan pengiriman data

Membaca Inputan Pesan Client dan mengirimkan ke Server

Penanganan Kesalahan (Error Handling)

Server :

```java
import java.io.*;
import java.net.*;
import java.net.ServerSocket;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Server {
    private static List<Socket> clientSockets = new ArrayList<>();
    private static List<ClientHandler> clientHandlers = new ArrayList<>();

    public static void main(String[] args) {
        try {
            // Membuat ServerSocket yang mendengarkan koneksi pada port 12345
            ServerSocket serverSocket = new ServerSocket(12345);
            System.out.println("Server berjalan, menunggu koneksi...");

            while (true) {
                // Menerima koneksi dari klien
                Socket clientSocket = serverSocket.accept();

                // Menerima nama klien saat terhubung
                BufferedReader reader = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
                String clientName = reader.readLine();
                System.out.println(clientName +" " + "Terhubung ke server.");

                if (isNameUnique(clientName)) {
                    clientSockets.add(clientSocket);
                    // Menambahkan socket klien dan handler ke daftar
                    ClientHandler clientHandler = new ClientHandler(clientSocket, clientName);
                    clientHandlers.add(clientHandler);
                    // Memulai thread untuk menangani klien
                    Thread clientThread = new Thread(clientHandler);
                    clientThread.start();
                } else {
                    // Jika nama klien tidak unik, beri tahu klien dan tutup koneksi
                    PrintWriter writer = new PrintWriter(clientSocket.getOutputStream(), true);
                    writer.println("Nama klien telah digunakan. Silakan coba dengan nama lain.");

                    // Menutup socket client
                    clientSocket.close();
                }

                // Mulai thread untuk mengirim pesan ke semua klien
                Thread senderThread = new Thread(new MessageSender());
                senderThread.start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void broadcastMessage(String message) {
        synchronized (clientHandlers) {
            // Mengirim pesan ke semua klien yang terhubung
            for (ClientHandler handler : clientHandlers) {
                handler.sendMessage(message);
            }
        }
    }
    private static boolean isNameUnique(String name) {
        synchronized (clientHandlers) {
            // Memeriksa apakah nama klien unik
            for (ClientHandler handler : clientHandlers) {
                if (handler.getClientName().equals(name)) {
                    return false;
                }
            }
        }
        return true;
    }
}

class ClientHandler implements Runnable {
    private Socket clientSocket;
    private BufferedReader in;
    private PrintWriter out;
```

Server :

Inisialisasi dan Pembuatan Socket untuk mendengarkan koneksi pada port 12345

Pembuatan Array daftar List untuk Socket dan Client Handler

Pembuatan Aliran penerimaan pesan & pembacaan pesan yang diterima dari client saat pertama setelah terhubung berupa "nama client"

Multithreading : penambahan socket client & client handler ke list

Perintah opsi lain apabila "nama Client" yang diterima tidak unik/sudah terdaftar makan akan ditolak

Thread all Client

Penanganan Kesalahan (Error Handling)

Method Broadcast pesan ke semua client

Method Penanganan client yang telah terhubung pada awal, dengan cek karakter nama client yang unik

```
80      private String clientName;
81      //untuk menangani klien
82      public ClientHandler(Socket socket, String clientName) {
83          this.clientSocket = socket;
84          this.clientName = clientName;
85      }
88      public void run() {
89          try {
90              //untuk membuka dan menutup stream
91              in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
92              out = new PrintWriter(clientSocket.getOutputStream(), true);
93
94              String message;
95              //untuk menampikan pesan dari klien
96              while ((message = in.readLine()) != null) {
97                  System.out.println("Pesan dari " + clientName + ": " + message);
98
99                  // Mengirim pesan ke semua klien
100                 Server.broadcastMessage("Pesan dari " + clientName + ": " + message);
101             }
102             in.close();
103             out.close();
104             clientSocket.close();
105         } catch (IOException e) {
106             System.out.println("Client " + clientName + ": Disconnect");
107         }
108     }
109
110     public void sendMessage(String message) {
111         //untuk mengirim pesan kepada klien
112         out.println(message);
113     }
114
115     public String getClientName() {
116         //untuk mendapatkan nama klien
117         return clientName;
118     }
119 }
121 class MessageSender implements Runnable {
122     @Override
123     public void run() {
124         while (true) {
125             // Baca pesan dari input (console) server
126             Scanner scanner = new Scanner(System.in);
127             System.out.print("");
128             String message = scanner.nextLine();
129
130             // Mengirim pesan ke semua klien
131             Server.broadcastMessage("Server: " + message);
132         }
133     }
134 }
```

Pembuatan I/O Stream untuk komunikasi dengan client

Menampilkan pesan dari client

Mengirimkan pesan ke semua Client terhubung

Perintah tutup stream & socket

Penanganan Kesalahan (Error Handling)

Method untuk mengirim pesan ke Client

Method untuk mendapatkan "nama Client" yang telah dikirim

Kelas untuk mengatur perintah membaca pesan yang diinput server dan mengirimnya ke semua Client