LAPORAN UJIAN TENGAH SEMESTER
JARINGAN KOMPUTER

Dosen Penganpu:

Taufiqotul Bariyah, S.Kom., M.IM., MCE

Disusun Oleh:

Ari Setia Hinanda (3012310005)

Muhammad Muqoffin Nuha (3012310023)

IF-3B

DEPARTEMEN INFORMATIKA

UNIVERSITAS INTERNASIONAL SEMEN INDONESIA

GRESIK

2024

Implementasi Aplikasi Chat Berbasis

Socket Programming Menggunakan Java

1. Pendahuluan

- **Latar Belakang**

  Komunikasi real-time melalui jaringan komputer adalah salah satu fondasi utama teknologi modern. Socket programming menggunakan Java memungkinkan pengembangan aplikasi yang efisien untuk komunikasi jaringan, termasuk aplikasi chat.

- **Tujuan**

  Tujuan dibuatnya aplikasi ini adalah guna memenuhi penugasan Ujian Tengah Semester yaitu membuat aplikasi chat berbasis socket programming dengan kemampuan multi-client untuk mendukunng komunikasi pesan teks di dalam jaringan lokal (LAN).

2. Arsitektur Sistem

2.1 Batasan Aplikasi

- **Server:**
  - Server berjalan pada alamat IP dan port yang ditentukan.
  - Server harus mampu menangani lebih dari satu client secara bersamaan menggunakan multithreading.
  - Ketika menerima pesan dari satu client, server harus mengirimkan pesan tersebut ke semua client yang terhubung.
  - Server harus mencatat setiap koneksi, pesan yang diterima, dan pesan yang dikirim pada konsol.
- **Client:**
  - Client terhubung ke server menggunakan alamat IP dan port yang ditentukan.
  - Client harus memiliki antarmuka pengguna sederhana dengan:
    - Kotak teks untuk menampilkan pesan yang diterima.
    - Kotak input untuk menulis pesan yang akan dikirim.

- Tombol untuk mengirim pesan.
- Ketika client mengirimkan pesan, pesan tersebut harus muncul di client lain yang terhubung.

2.2 Komponen Utama

- **ChatServer**: Mengelola koneksi dan distribusi pesan
- **ChatClient**: Antarmuka pengguna untuk mengirim dan menerima pesan

2.3 Teknologi Yang Digunakan

- **Bahasa Pemrograman:** Java
- **Socket Programming:** java.net
- **Antarmuka Grafis:** Java Swing
- **Fitur Multi-threading:** Java Thread API

3. Implementasi Teknis

a. Server (ChatServer.java)

i. Umum:

- Port default: 12345
- Mendukung koneksi multi-client
- Broadcast pesan ke semua client

ii. Struktur Kode Utama

```
public class ChatServer {
    // Port default untuk koneksi
    private static final int PORT = 12345;

    // Kumpulan writer untuk semua client yang terhubung
    private static HashSet<PrintWriter> clientWriters = new HashSet<>();

    public static void main(String[] args) {
        // Metode utama untuk menjalankan server
    }

    // Kelas inner untuk menangani koneksi setiap client
    private static class ClientHandler implements Runnable {
        // Implementasi logika komunikasi per client
    }
}
```

iii. Multi-Threading

1. ServerSocket Listener

```java
try (ServerSocket serverSocket = new ServerSocket(PORT)) {
    while (true) {
        // Menunggu dan menerima koneksi client
        Socket clientSocket = serverSocket.accept();

        // Setiap koneksi baru dibuat thread terpisah
        ClientHandler handler = new ClientHandler(clientSocket);
        new Thread(handler).start();
    }
}
```

- *serverSocket.accept()* memblokir dan menunggu koneksi baru
- Setiap koneksi menghasilkan thread terpisah
- Memungkinkan komunikasi paralel dengan multiple client

2. Manajemen Koneksi Client

```java
public void run() {
    try {
        // Inisialisasi stream input/output
        in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        out = new PrintWriter(socket.getOutputStream(), true);

        // Tambahkan writer client ke kumpulan writer
        synchronized (clientWriters) {
            clientWriters.add(out);
        }

        // Loop membaca pesan
        String message;
        while ((message = in.readLine()) != null) {
            System.out.println("Received: " + message);
            broadcast(message);
        }
    } catch (IOException e) {
        System.err.println("Error handling client: " + e.getMessage());
    } finally {
        // Pembersihan sumber daya
        synchronized (clientWriters) {
            clientWriters.remove(out);
        }
        try {
            socket.close();
        } catch (IOException e) {
            System.err.println("Error closing socket: " + e.getMessage());
        }
    }
}
```

3. Mekanisme Broadcast

```
private void broadcast(String message) {
    synchronized (clientWriters) {
        // Mengirim pesan ke semua client yang terhubung
        for (PrintWriter writer : clientWriters) {
            writer.println(message);
        }
    }
}
```

- Menggunakan *synchronized* untuk thread-safety
- Menjamin pesan dikirim ke semua client

b. Client (ChatClient.java

i. Umum:

- Konfigurasi koneksi dinamis
- Antarmuka grafis dengan format pesan berbeda
- Manajemen koneksi dan error handlingclient

ii. Arsitektur Antarmuka Pengguna

```
public class ChatClient {
    // Komponen GUI
    private JFrame frame;
    private JTextPane messageArea;
    private JTextField messageInput;

    // Koneksi jaringan
    private BufferedReader in;
    private PrintWriter out;
    private String serverIP, username;
}
```

iii. Konfigurasi Dinamis

```
private boolean showConfigDialog() {
    JPanel panel = new JPanel(new GridLayout(0, 1));

    // Input fields untuk konfigurasi
    JTextField ipField = new JTextField("localhost", 20);
    JTextField portField = new JTextField("12345", 20);
    JTextField usernameField = new JTextField(20);

    // Validasi input
    if (result == JOptionPane.OK_OPTION) {
        try {
            String ip = ipField.getText().trim();
            int port = Integer.parseInt(portField.getText().trim());
            String username = usernameField.getText().trim();

            // Pemeriksaan input
            if (ip.isEmpty() || username.isEmpty()) {
                JOptionPane.showMessageDialog(null,
                    "Semua field harus diisi.",
                    "Kesalahan Input",
                    JOptionPane.ERROR_MESSAGE);
                return false;
            }

            // Validasi port
            if (port < 1 || port > 65535) {
                JOptionPane.showMessageDialog(null,
                    "Port harus antara 1 dan 65535.",
                    "Kesalahan Input",
                    JOptionPane.ERROR_MESSAGE);
                return false;
            }

            // Set konfigurasi
            this.serverIP = ip;
            this.serverPort = port;
            this.username = username;
            return true;
        } catch (NumberFormatException e) {
            // Penanganan kesalahan format port
            JOptionPane.showMessageDialog(null,
                "Nomor port tidak valid.",
                "Kesalahan Input",
                JOptionPane.ERROR_MESSAGE);
            return false;
        }
    }
    return false;
}
```

iv. Koneksi dan Penerimaan Pesan

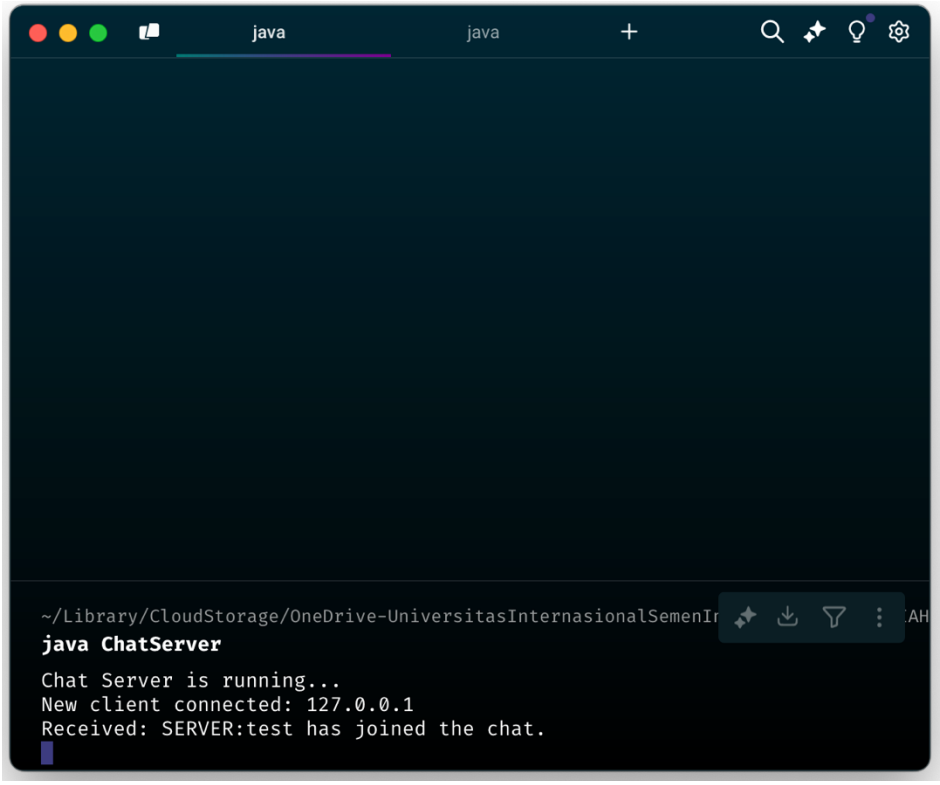
```
private void connectToServer() throws IOException {
    // Membuat koneksi socket
    Socket socket = new Socket(serverIP, serverPort);
    in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
    out = new PrintWriter(socket.getOutputStream(), true);

    // Mengirim pesan bergabung
    out.println("SERVER:" + username + " has joined the chat.");

    // Thread terpisah untuk menerima pesan
    new Thread(() -> {
        try {
            String message;
            while ((message = in.readLine()) != null) {
                // Memproses pesan berdasarkan jenisnya
                if (message.startsWith("SERVER:")) {
                    appendMessage(message.substring(7), "centerStyle");
                } else if (message.startsWith(username + ": ")) {
                    appendMessage(message.substring(username.length() + 2), "rightStyle");
                } else {
                    appendMessage(message, "leftStyle");
                }
            }
        } catch (IOException e) {
            appendMessage("Koneksi ke server terputus", "centerStyle");
        }
    }).start();
}
```

v. Formatting Pesan

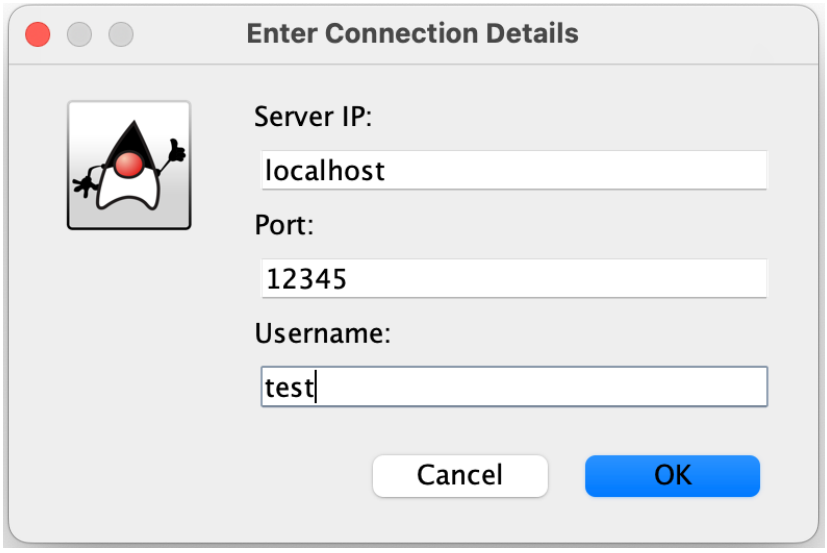
```
private void addStylesToDocument() {
    // Membuat style untuk berbagai jenis pesan
    Style leftStyle = messageArea.addStyle("leftStyle", null);
    StyleConstants.setAlignment(leftStyle, StyleConstants.ALIGN_LEFT);
    StyleConstants.setForeground(leftStyle, Color.BLACK);

    Style rightStyle = messageArea.addStyle("rightStyle", null);
    StyleConstants.setAlignment(rightStyle, StyleConstants.ALIGN_RIGHT);

    Style centerStyle = messageArea.addStyle("centerStyle", null);
    StyleConstants.setAlignment(centerStyle, StyleConstants.ALIGN_CENTER);
    StyleConstants.setForeground(centerStyle, Color.GRAY);
    StyleConstants.setItalic(centerStyle, true);
}
```

4. Pengujian

- **Lingkungan Pengujian**
  - **Hardware:** Laptop.
  - **Software:** JDK 21, Visual Studio Code.
  - **Jaringan:** LAN menggunakan koneksi Wi-Fi lokal.
- **Running Server**



- **Running Client**
  - **Inisialisasi (IP Server, Port, Username) secara dinamis**



  - **Tampilan Forum**

5. **Kesimpulan**
   - Aplikasi chat berbasis socket programming berhasil dibuat dan berfungsi sesuai spesifikasi.
   - Server mendukung komunikasi multi-client dengan stabilitas yang baik.
   - Antarmuka pengguna sederhana namun efektif untuk pengiriman dan penerimaan pesan.

6. **Lampiran**
   1. **Source Code Full : https://github.com/initheo/UTS-Jarkom-2024**
   2. **Persyaratan Sistem**
      a. Java Development Kit (JDK) 11 atau lebih tinggi
      b. Koneksi jaringan local/internet
   3. **Konfigurasi Port**

a. Default: 12345

b. Dapat diubah di ChatServer.java dan ChatClient.java