

## **Tugas Rumah**

Tugas ini bertujuan untuk memberikan pemahaman dan pengalaman praktis dalam mengimplementasikan pemrograman jaringan menggunakan model stream (TCP) dan datagram (UDP).

### **Instruksi: Bagian 1: Model Stream (TCP)**

1. Server TCP (FileServerTCP.java):

- Implementasikan server TCP yang dapat menerima file dari klien dan menyimpannya di server.
- Server harus dapat menangani multiple klien secara bersamaan.
- Berikan informasi kepada klien setiap kali file berhasil diterima.

2. Klien TCP (FileClientTCP.java):

- Implementasikan klien TCP yang dapat mengirim file ke server.
- Klien harus dapat menangani pengiriman file dari path yang ditentukan oleh pengguna.

3. Uji Coba:

- Jalankan beberapa klien sekaligus untuk menguji kemampuan server menangani multiple klien.
- Kirimkan file dari beberapa klien ke server secara bersamaan dan periksa apakah file tersebut berhasil diterima.

### **Bagian 2: Model Datagram (UDP)**

1. Server UDP (ChatServerUDP.java):

- Implementasikan server UDP yang dapat menerima pesan dari klien dan menyebarkannya ke semua klien yang terhubung.
- Server harus dapat menangani multiple klien secara bersamaan.
- Pesan yang diterima oleh server harus mencakup informasi pengirim dan kontennya.

2. Klien UDP (ChatClientUDP.java):

- Implementasikan klien UDP yang dapat mengirim pesan ke server.
- Klien harus dapat menangani pengiriman pesan ke server.

3. Uji Coba:

- Jalankan beberapa instansi klien sekaligus untuk menguji kemampuan server menangani multiple klien.

Tugas Modul 4 – Stream dan Datagram  
Muhammad Maksum (362241009)  
Salman Al Majali (362241010)  
Inov Whily (362389005)  
Agung Wicaksono (362389010)

---

- Kirimkan pesan dari salah satu klien dan periksa apakah pesan tersebut disebarkan ke semua klien yang terhubung.

**Jawaban:**

**TCP – Server**

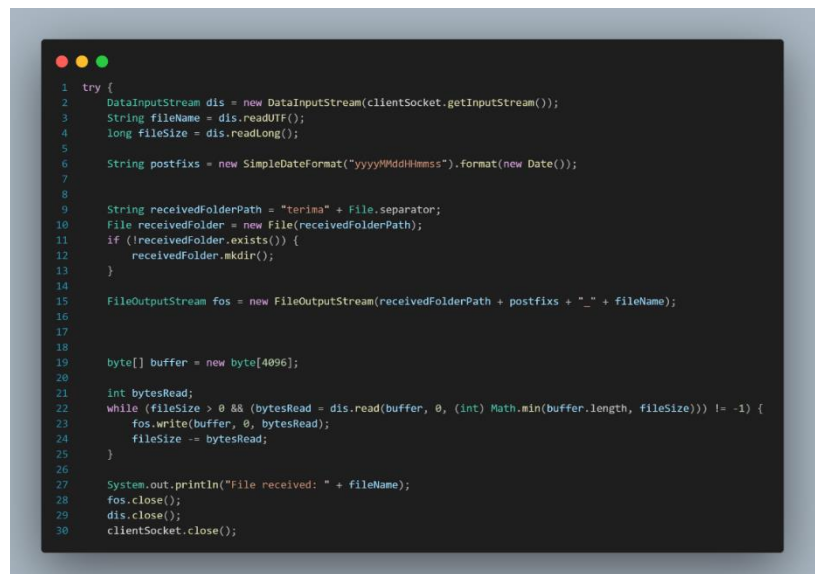


```
1 try {
2     ServerSocket serverSocket = new ServerSocket(12345);
3     System.out.println("Server is running. Waiting for clients...");
```

Pada saat ini, server akan membuka socket menggunakan ServerSocket pada port 12345 dan akan mencetak “Server is running. Waiting for clients...”. Akan dilakukan looping untuk menerima client, sehingga setiap kali terdapat client yang terhubung, server akan membuat objek FileHandler baru dan menjalankannya dalam thread baru.




```
1 while (true) {
2     Socket clientSocket = serverSocket.accept();
3     new Thread(new FileHandler(clientSocket)).start();
4 }
```



```
1 try {
2     DataInputStream dis = new DataInputStream(clientSocket.getInputStream());
3     String fileName = dis.readUTF();
4     long fileSize = dis.readLong();
5
6     String postfixs = new SimpleDateFormat("yyyyMMddHHmmss").format(new Date());
7
8
9     String receivedFolderPath = "terina" + File.separator;
10    File receivedFolder = new File(receivedFolderPath);
11    if (!receivedFolder.exists()) {
12        receivedFolder.mkdir();
13    }
14
15    FileOutputStream fos = new FileOutputStream(receivedFolderPath + postfixs + "_" + fileName);
16
17
18    byte[] buffer = new byte[4096];
19
20    int bytesRead;
21    while (fileSize > 0 && (bytesRead = dis.read(buffer, 0, (int) Math.min(buffer.length, fileSize))) != -1) {
22        fos.write(buffer, 0, bytesRead);
23        fileSize -= bytesRead;
24    }
25
26
27    System.out.println("File received: " + fileName);
28    fos.close();
29    dis.close();
30    clientSocket.close();
```

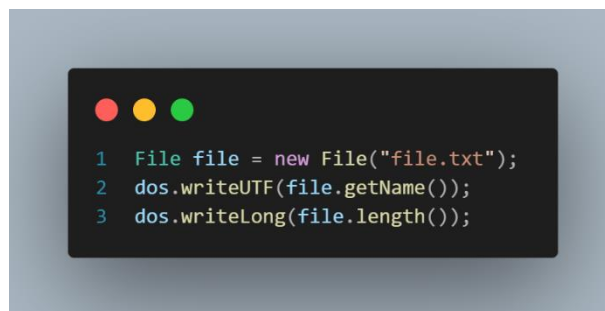
FileHandler akan mengeksekusi dalam thread terpisah, sehingga dalam metode run() akan menerima input dari klien mengenai nama file dan ukuran dokumen, kemudian membuat folder penerima jika belum ada, membuat objek stream untuk menulis file yang diterima, dan membaca file byte per byte dari DataInputStream serta menuliskannya ke file yang sedang dibuka. Setelah selesai, menutup FileOutputStream, FileInputStream, dan socket client.

## TCP – Client



```
1 try {  
2     Socket socket = new Socket("localhost", 12345);  
3     DataOutputStream dos = new DataOutputStream(socket.getOutputStream());
```

Pada saat ini, klien akan membuat koneksi socket dengan server yang berjalan di localhost (127.0.0.1) pada port 12345.



```
1 File file = new File("file.txt");  
2 dos.writeUTF(file.getName());  
3 dos.writeLong(file.length());
```

Klien membuka file "file.txt" untuk dibaca kemudian mengirimkan nama file dan ukuran file ke server menggunakan DataOutputStream.



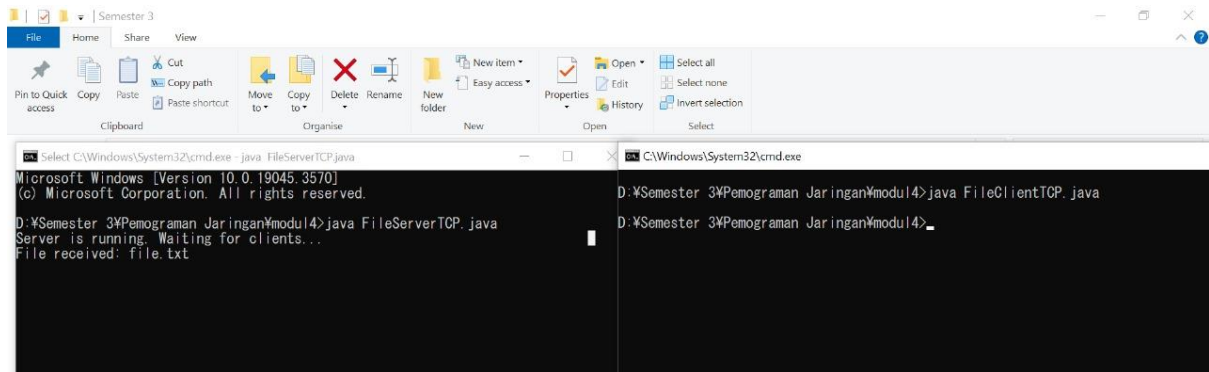
```
1 FileInputStream fis = new FileInputStream(file);  
2 byte[] buffer = new byte[4096];  
3  
4 int bytesRead;  
5 while ((bytesRead = fis.read(buffer)) != -1) {  
6     dos.write(buffer, 0, bytesRead);  
7 }
```

Klien membuka input stream dari file dan membaca file dalam potongan-potongan (buffer) sebesar 4096 byte. Setiap potongan dibaca, klien mengirimkannya ke server melalui output stream.

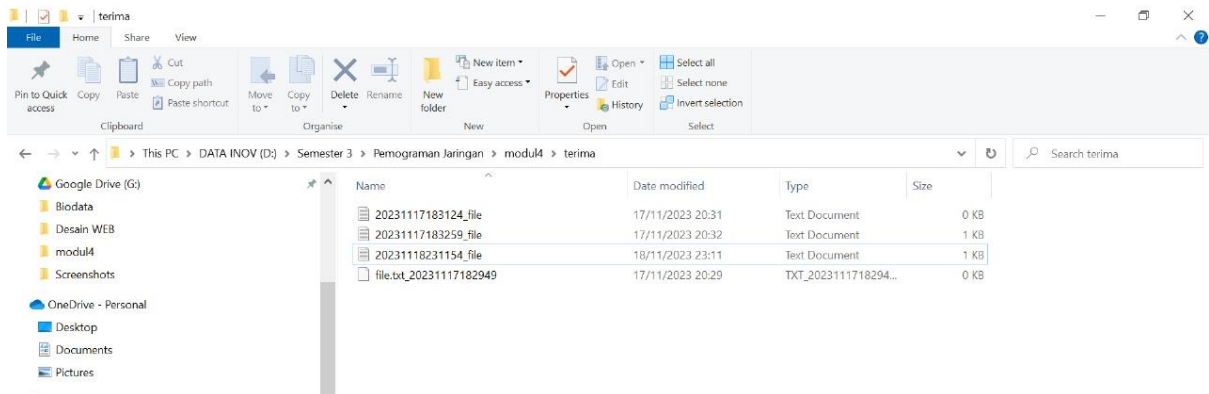
Tugas Modul 4 – Stream dan Datagram  
Muhammad Maksum (362241009)  
Salman Al Majali (362241010)  
Inov Whily (362389005)  
Agung Wicaksono (362389010)

---

Berikut adalah tampilan program TCP yang kami buat saat dijalankan:



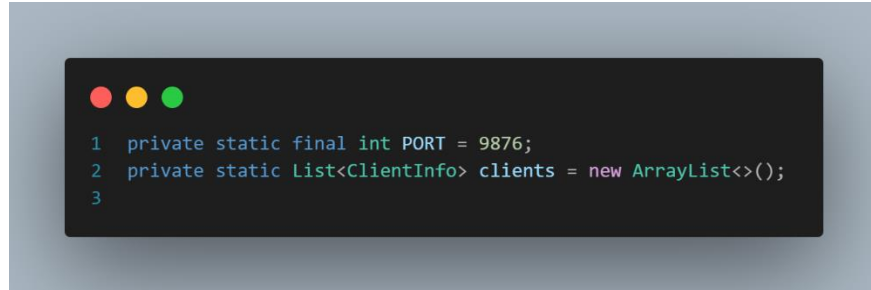
Saat dijalankan, client dapat mengirim file ke server dan server akan menerima dokumen yang telah dikirimkan oleh client. Dokumen yang telah diterima server akan disimpan dalam folder “terima” yang telah dibuat seperti gambar dibawah ini.



Tugas Modul 4 – Stream dan Datagram  
Muhammad Maksum (362241009)  
Salman Al Majali (362241010)  
Inov Whily (362389005)  
Agung Wicaksono (362389010)

---

## UDP – Server



Pada program UDP yang kami buat, menggunakan kelas dari package java.io dan java.net untuk berkomunikasi melalui jaringan menggunakan UDP. Server menggunakan port 9876 untuk mendengarkan koneksi, dan daftar klien disimpan dalam list clients.



Pada method main ini, server akan membuat socket UDP dan fokus pada port yang ditentukan. Program akan melakukan looping dan akan terus menerima sambungan dari klien. Setiap package yang diterima diubah menjadi string dan informasi pengirimnya diambil. Pesan yang diterima kemudian ditampilkan di server. Pesan tersebut kemudian dibagikan ke semua klien kecuali pengirim asli.

Tugas Modul 4 – Stream dan Datagram  
Muhammad Maksum (362241009)  
Salman Al Majali (362241010)  
Inov Whily (362389005)  
Agung Wicaksono (362389010)

---

```
1 private static void broadcastMessage(Message message, String senderAddress, int senderPort) {  
2     for (ClientInfo client : clients) {  
3         if (!client.getAddress().equals(senderAddress) && client.getPort() == senderPort) {  
4             sendToClient(message, client.getAddress(), client.getPort());  
5         }  
6     }  
7 }  
8
```

Code tersebut merupakan perintah untuk mengirim pesan ke semua klien yang tersambung kecuali pengirim.

```
1 private static void sendToClient(Message message, String address, int port) {  
2     try {  
3         // Membuat socket dan output stream  
4         DatagramSocket socket = new DatagramSocket();  
5         ByteArrayOutputStream outputStream = new ByteArrayOutputStream();  
6         ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);  
7  
8         // Mengubah pesan menjadi byte array  
9         objectOutputStream.writeObject(message);  
10        byte[] sendData = outputStream.toByteArray();  
11  
12        // Membuat paket dan mengirimnya ke klien  
13        DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,  
14            java.net.InetAddress.getByName(address), port);  
15        socket.send(sendPacket);  
16        socket.close();  
17    } catch (Exception e) {  
18        e.printStackTrace();  
19    }  
20 }  
21
```

Pada method sendToClient, code tersebut adalah perintah untuk mengirim pesan objek message ke klien dengan alamat dan port yang ditentukan.

Tugas Modul 4 – Stream dan Datagram  
Muhammad Maksum (362241009)  
Salman Al Majali (362241010)  
Inov Whily (362389005)  
Agung Wicaksono (362389010)

---

```
1 private static void sendToClient(Message message, String address, int port) {
2     try {
3         // Membuat socket dan output stream
4         DatagramSocket socket = new DatagramSocket();
5         ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
6         ObjectOutputStream objectOutputStream = new ObjectOutputStream(outputStream);
7
8         // Mengubah pesan menjadi byte array
9         objectOutputStream.writeObject(message);
10        byte[] sendData = outputStream.toByteArray();
11
12        // Membuat paket dan mengirimnya ke klien
13        DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,
14            java.net.InetAddress.getByName(address), port);
15        socket.send(sendPacket);
16        socket.close();
17    } catch (Exception e) {
18        e.printStackTrace();
19    }
20 }
21
```

Code tersebut akan digunakan untuk menyimpan informasi klien (address dan port).

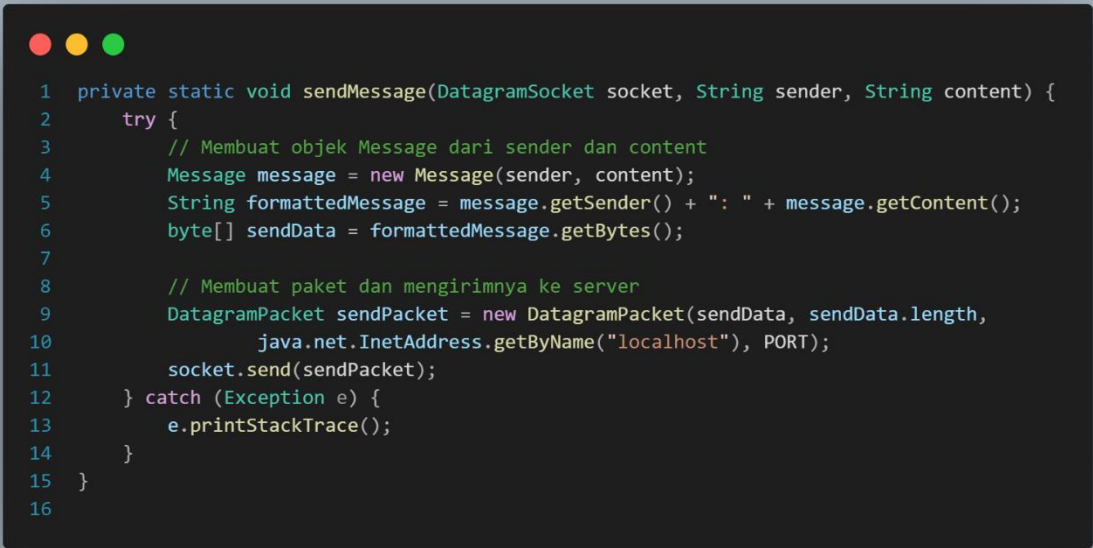
## UDP – Client

```
1 private static final int PORT = 9876;
```

Pada program UDP yang kami buat, klien menggunakan port 9876 untuk terhubung dengan server.

```
1 public static void main(String[] args) {
2     try {
3         DatagramSocket clientSocket = new DatagramSocket();
4         Scanner scanner = new Scanner(System.in);
5
6         System.out.print("Enter your name: ");
7         String name = scanner.nextLine();
8
9         while (true) {
10            System.out.print("[ " + name + " ] Enter message or file path (type 'exit' to quit): ");
11            String input = scanner.nextLine();
12
13            if ("exit".equalsIgnoreCase(input)) {
14                break;
15            }
16
17            // Kirim pesan
18            sendMessage(clientSocket, name, input);
19        }
20
21        clientSocket.close();
22        scanner.close();
23    } catch (Exception e) {
24        e.printStackTrace();
25    }
26 }
27
```

Code tersebut merupakan method main yang merupakan code perintah agar klien membuat socket UDP dan klien akan diminta untuk menginput nama. Kemudian program akan melakukan looping, kemudian klien akan meminta pengguna untuk memasukkan pesan atau path dokumen. Apabila user menginput exit, program akan berhenti melakukan looping. Selama looping dilakukan, klien akan menggunakan method sendMessage untuk mengirim pesan ke server.



```
1 private static void sendMessage(DatagramSocket socket, String sender, String content) {  
2     try {  
3         // Membuat objek Message dari sender dan content  
4         Message message = new Message(sender, content);  
5         String formattedMessage = message.getSender() + ": " + message.getContent();  
6         byte[] sendData = formattedMessage.getBytes();  
7  
8         // Membuat paket dan mengirimnya ke server  
9         DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length,  
10             java.net.InetAddress.getByName("localhost"), PORT);  
11         socket.send(sendPacket);  
12     } catch (Exception e) {  
13         e.printStackTrace();  
14     }  
15 }  
16
```

Code tersebut merupakan method sendMessage yang merupakan perintah untuk membuat objek pesan dari pengirim dan isi pesan. Pesan akan diberi nama dengan format “nama pengirim: isi pesan” dan dikonversi menjadi byte array. Kemudian package datagram dibuat dan dikirim ke server.