Institut Teknologi Del

10S2202 Jaringan Komputer
Semester Genap 2021/2022

Praktikum Week 12 Socket Programming I

Bagian I Single Client, Single Server

Pada bagian ini, kita akan membuat sebuah aplikasi client server sederhana menggunakan java socket programming.
Aplikasi yang akan dibuat memiliki fungsi yang sederhana:

1. Client akan terhubung ke server pada port yang telah ditentukan oleh server dan juga memberikan id unik dari client (nama client)
2. Client akan membaca sebuah string yang akan diketikkan oleh user
3. Client akan mengirimkan string tersebut ke server
4. Server akan menerima string tersebut mengubahnya menjadi huruf besar dan mengirimkannya kembali ke client.
5. Server juga akan menampilkan informasi string yang diterima dari client
6. Client akan menuliskan string yang diterima ke layar
7. Operasi mengirim dan menerima string tersebut akan berjalan terus menerus dan diakhiri ketika client mengirimkan string “quit”

Pertama, kita akan membuat program client. Buatlah sebuah file java **TCPClient.java** dengan kerangka program berikut:

Note: Jangan biasakan copy paste program, silahkan tulis ulang supaya lebih mengerti.

```
import java.io.*;
import java.net.*;

class TCPClient {
	public static void main(String argv[]) throws Exception {
		String sentence;
		String modifiedSentence;

		if (argv.length < 3) {
			System.out.println("Jalankan program dengan memberikan
parameter IP Server, Port, dan nama client\n");
			System.out.println("Contoh:");
			System.out.println("java TCPClient 127.0.0.1 8999
budi");
			System.exit(0);
```

```
            }
            //tambahkan code disini
      }
}
```

Kode program diatas berisi kerangka program yang dilengkapi dengan validasi input dari client.
Compile file tersebut dengan perintah **#javac TCPClient.java**, jalankan dengan perintah **java TCPClient**, pesan error akan muncul beserta informasi cara menjalankan program dengan benar Cth: java TCPClient 127.0.0.1 8999 (127.0.0.1 adalah IP address localhost).

Selanjutnya, kita akan melengkapi program dengan kode program untuk melakukan pembentukan socket dan melakukan koneksi TCP ke server.
Tambahkan kode program berikut ke bagian yang telah ditandai (//tambahkan code disini).

```
SocketAddress socketAddr = new InetSocketAddress(argv[0],
Integer.parseInt(argv[1]));
int timeout = 2000; //millisecond
Socket clientSocket = new Socket();
DataOutputStream outToServer = null;
BufferedReader inFromServer = null;
try{
      System.out.println("Connecting to server on IP: " + argv[0] + "on
PORT: " + argv[1]);
      clientSocket.connect(socketAddr, timeout);
      System.out.println("Terhubung ke server !!");
      outToServer = new DataOutputStream(clientSocket.getOutputStream());
      inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));

      outToServer.writeBytes(argv[2] + '\n');
      modifiedSentence = inFromServer.readLine();
      System.out.println("FROM SERVER: " + modifiedSentence);
} catch (SocketTimeoutException exception) {
      System.out.println("SocketTimeoutException " + argv[0] + ":" +
argv[1] + ". " + exception.getMessage());
      System.exit(0);
} catch (IOException exception) {
      System.out.println("IOException - Unable to connect to " + argv[0]
+ ":" + argv[1] + ". " + exception.getMessage());
      System.exit(0);
}

BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
System.out.println("Tuliskan \"quit\" untuk berhenti !!");
do {
      System.out.print("Input String: ");
      sentence = inFromUser.readLine();
      outToServer.writeBytes(sentence + '\n');
      modifiedSentence = inFromServer.readLine();
      System.out.println("FROM SERVER: " + modifiedSentence);
}while(!sentence.toUpperCase().equals("QUIT"));

clientSocket.close();
```

Berikut ini adalah penjelasan bagian-bagian penting dari program.

```
SocketAddress socketAddr = new InetSocketAddress(argv[0],
Integer.parseInt(argv[1]))
```

Kode program ini berfungsi untuk membuat sebuah variable yang berfungsi untuk menampung informasi ip address server beserta port TCP yang akan dihubungi. Informasi tersebut diterima sebagai parameter dalam menjalankan program (argv[0] dan argv[1]).

```
clientSocket.connect(socketAddr, timeout);
```

Kode program ini berfungsi untuk melakukan pembentukan koneksi TCP ke server yang dilengkapi dengan batasan waktu yang dibutuhkan dalam membentuk koneksi.

```
outToServer = new DataOutputStream(clientSocket.getOutputStream());
inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
```

Bagian kode program ini berfungsi untuk membentuk variable yang bias kita gunakan untuk membaca data dari socket dan menuliskan data ke socket (nama variable telah disesuaikan).

Silahkan anda compile ulang program tersebut dan jalankan.

Program tersebut tentunya tidak akan bisa terhubung ke server karena server penerima koneksi belum ada.

Sekarang kita akan membuat aplikasi server.
Buatlah sebuah file java dengan nama **TCPServer.java** yang berisi kerangka program berikut

```
import java.io.*;
import java.net.*;
class TCPServer {
	public static void main(String argv[]) throws Exception {
		String clientSentence;
		String capitalizedSentence;

		if (argv.length != 1) {
			System.out.println("Jalankan server dengan memberikan
argumen port");
		System.err.println("Contoh: java TCPServer 8999");
		System.exit(1);
	}

		//tambahkan code disini
	}
}
```

Sama seperti aplikasi client, kerangka dasar ini masih berisikan validasi input dari user untuk menjalankan program dengan benar. Program harus dijalankan dengan memberikan parameter berupa nomor PORT yang akan digunakan Cth. Java TCPServer 8999.

Selanjutnya kita akan menambahkan kode program yang berfungsi untuk membentuk PORT TCP dan me-listen koneksi dari client.
Silahkan tambahkan kode program berikut pada bagian yang telah ditandai

```
try(ServerSocket welcomeSocket = new
ServerSocket(Integer.parseInt(argv[0]))) {
	while(true) {
		Socket connectionSocket = welcomeSocket.accept();
		BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
```

```
            DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
            //baca greetings
            String namaClient = inFromClient.readLine();
            System.out.println("Client " + namaClient + " Terhubung");
            outToClient.writeBytes("Horas " + namaClient + '\n');
            while((clientSentence = inFromClient.readLine()) != null){
                  System.out.println("dari " + namaClient + " : " +
clientSentence);
                  capitalizedSentence = clientSentence.toUpperCase() +
'\n';
                  outToClient.writeBytes(capitalizedSentence);
                  if (capitalizedSentence.equals("QUIT\n")) {
                        System.out.println(namaClient + " telah
berhenti");
                        break;
                  }
            }
            connectionSocket.close();

      }
} catch (IOException e) {
    System.err.println("Server tidak bisa menggunakan port " + argv[0]);
    System.err.println("Port tersebut kemungkinan sudah digunakan !!");
    System.exit(-1);
}
```

Berikut ini adalah penjelasan bagian penting dari program

```
try(ServerSocket welcomeSocket = new
ServerSocket(Integer.parseInt(argv[0]))) {} catch (..){}
```

Bagian ini berfungsi untuk mencoba membentuk sebuah socket server dengan nomor port yang diberikan oleh user. Program akan error dan memberikan informasi error jika nomor port yang diberikan oleh user tidak bisa digunakan (mis. Sudah digunakan oleh program yang lain).

```
Socket connectionSocket = welcomeSocket.accept();
BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
```

Bagian program ini berfungsi untuk me-listen koneksi dari client dan menerima koneksi jika ada dan dilanjutkan dengan membentuk variable yang akan digunakan untuk memaca dan menerima data dari socket.

Silahkan Compile file java anda.
Untuk mencoba program, jalankanlah aplikasi dengan memberikan informasi port yang telah digunakan oleh program lain (mis. 80). Program akan berhenti dan memberikan informasi error. Hal ini dikarenakan sifat dari socket, bahwa sebuah nomor port hanya bisa digunakan oleh 1 program pada sebuah host (IP Address) tertentu.

Note: Gunakan perintah **netstat** atau aplikasi **cports** (windows only) yang disertakan dalam praktikum ini untuk melihat daftar port yang terbuka pada computer anda.

Selanjutnya, jalankan server anda dengan memberikan parameter yang benar.

Cth. **Java TCPServer 8999.**

Pada teriminal yang berbeda, jalankanlah program client dengan memberikan informasi server yang benar
Cth. **Java TCPClient 127.0.0.1 8999.**

Anda akan melihat bahwa client dapat terhubung ke server dan dapat bertukar informasi.

Selanjutnya, kita sebuah client terhubung ke server, bukalah sebuah terminal baru dan jalankan aplikasi client dan berikan informasi server yang sama.

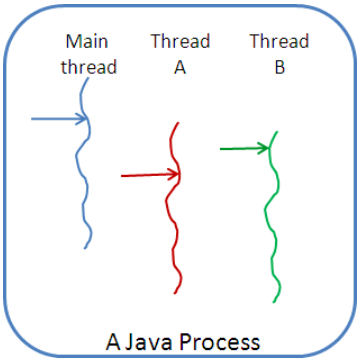
Anda akan melihat kalau client yang ke-2 tidak bisa terhubung ke server (tidak bisa dilayani oleh server) hal ini dikarenakan socket yang digunakan oleh server masih digunakan untuk melayani client yang lain (client yang pertama).

Untuk itu kita akan mengubah program server supaya bisa menerima koneksi dari banyak client (Bagian II).

Bagian II Multi Client, Single Server

Untuk memungkinkan Server menerima koneksi dari beberapa client pada saat yang bersamaan, kita akan melengkapi aplikasi server kita dengan Thread.

Thread adalah sub-proses yang dapat berjalan secara mandiri tanpa mengganggu proses utama (masih ingat konsep resepsionis hotel?).



Program yang akan dibuat akan berjalan dengan alur berikut:

1. Server akan membentuk socket server yang akan dihubungi oleh client
2. Jika ada client yang mencoba melakukan koneksi, server akan menerima koneksi tersebut dan membentuk sebuah sub-proses (thread) untuk melayani client tersebut
3. Server utama akan kembali me-listen koneksi dari client yang lain dan langkah yang sama akan dilakukan (#2) jika ada koneksi baru.

Buatlah sebuah file java baru dengan nama **TCPServerThread.java** dengan kode program berikut:

```
import java.net.*;
import java.io.*;
```

```
public class TCPServerThread extends Thread {
    private Socket connectionSocket = null;

    public TCPServerThread(Socket socket) {
      super("TCPServerThread");
      this.connectionSocket = socket;
    }

    public void run(){
      try{
            String clientSentence;
                  String capitalizedSentence;
            BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
                  DataOutputStream outToClient = new
DataOutputStream(connectionSocket.getOutputStream());
                  //baca greetings
                  String namaClient = inFromClient.readLine();
                  System.out.println("Client " + namaClient + "
Terhubung");
                  outToClient.writeBytes("Horas " + namaClient + '\n');
                  while((clientSentence = inFromClient.readLine()) !=
null){
                        System.out.println("dari " + namaClient + " : " +
clientSentence);
                        capitalizedSentence =
clientSentence.toUpperCase() + '\n';
                        outToClient.writeBytes(capitalizedSentence);
                        if (capitalizedSentence.equals("QUIT\n")) {
                              System.out.println(namaClient + " telah
berhenti");
                              break;
                        }
                  }

                  connectionSocket.close();
            } catch (Exception e){
                  e.printStackTrace();
            }
    }
}
```

Kode program ini akan berfungsi sebagai thread (sub-proses) yang akan dijalankan oleh server utama. Kode program utama terdapat pada fungsi run() yang akan berfungsi untuk melayani client (menerima string, mengubah, dan mengirimkan hasil perubahan). Thread ini berjalan dengan menerima sebuah parameter berupa socket yang telah terhubung ke client. Socket tersebut diterima sebagai parameter dalam pembuatan thread (parameter constructor).

Selanjutnya untuk mengukana thread ini, kita akan membuat sebuah program server baru yang akan berfungsi untuk membentuk socket server (welcoming socket), listen koneksi, dan membentuk thread jika ada koneksi yang dating dari client.

Buatlah sebuah file java baru dengan nama **TCPServerMultiClient.java** yang berisi kode program berikut

```
import java.io.*;
import java.net.*;
```

```
class TCPServerMultiClient {
      public static void main(String argv[]) throws Exception {
            if (argv.length != 1) {
                  System.out.println("Jalankan server dengan memberikan
argumen port");
            System.err.println("Contoh: java TCPServer 8999");
            System.exit(1);
      }

            try(ServerSocket welcomeSocket = new
ServerSocket(Integer.parseInt(argv[0]))) {
                  while(true) {
                        new
TCPServerThread(welcomeSocket.accept()).start();
                  }
            } catch (IOException e) {
            System.err.println("Server tidak bisa menggunakan port " +
argv[0]);
            System.err.println("Port tersebut kemungkinan sudah digunakan
!!");
            System.exit(-1);
        }
      }
}
```

Bagian yang paling penting, jika dibandingkan dengan program server sebelumnya ada baris program berikut

```
while(true) {
      new TCPServerThread(welcomeSocket.accept()).start();
}
```

Bagian program ini berfungsi untuk me-listen koneksi dari client dan membentuk sebuah sub-proses (thread) jika koneksi tersebut berhasii diterima.

Compile file java yang baru dengan perintah **#javac TCPServerMultiClient.java**

Jalankanlah aplikasi server anda (java TCPServerMultiClient 8999) dan cobalah membentuk koneksi ke server dari beberapa client yang berbeda pada saat yang bersamaan.

TUGAS:
Implementasikanlah Bagian I dari praktikum ini dengan menggunakan socket UDP.
Perilaku aplikasi UDP anda harus sama dengan aplikasi TCP yang ada pada praktikum ini (perilaku input/output ke layar di client dan server).
Buatlah perbedaan antara UDP dengan TCP?
Bagaimana UDP dan TCP menangani kehilangan paket data?
Berikan pula contoh implementasi UDP dan TCP dalam aplikasi saat ini!

Selamat mengerjakan.