

LAPORAN PRAKTIKUM

0953633205  
  
PRAKTIKUM SISTEM TERDISTRIBUSI

MODUL: 3

“PEMROGRAMAN SOCKET DENGAN KONSEP MULTI-THREADING”

(Menggunakan Pemrograman Java)

NAMA : 1. Hakikid Amana (M0517019)

2. Mila Rosyida Uswatunnisa (M0517030)

3. Wafirotul Laila (M0517048)

KELOMPOK : 2

HARI, TANGGAL : JUMAT, 17 APRIL 2020

WAKTU : 12.45-15.00 WIB

ASISTEN : HAIDAR HENDRI SETYAWAN M0516023

TAUFIQ ODHI DWI PUTRA M0516043

PROGRAM STUDI INFORMATIKA

UNIVERSITAS SEBELAS MARET

2020

*Abstraksi Multi-thread* merupakan sebuah mekanisme di mana dalam suatu proses, ada beberapa thread yang mengerjakan tugasnya masing-masing pada waktu yang bersamaan. Supaya performansi atau load data lebih cepat maka diharuskan menggunakan *Multi- thread*. Dengan multi-thread semua gambar dapat terload secara bersamaan. Namun jika menggunakan *Single-thread*, akan membutuhkan waktu yang lama. Dalam percobaan ini dilakukan suatu pemrograman pada *socket* dengan konsep *multi-threading*. Percobaan dilakukan dengan membangun sebuah *socket programming* baik TCP dan UDPdengan *multi-threading* untuk proses penjumlahan menggunakan bahasa pemrograman java.

Modul 3

PEMROGRAMAN SOCKET DENGAN KONSEP MULTI-THREADING

(Menggunakan Pemrograman Java)

**HAKIKID AMANA** (M0517019), **MILA ROSYIDA USWATUNNISA** (M0517030), **WAFIROTUL LAILA** (M0517048) / Kelompok 2/ JUMAT, 17 APRIL 2020]

Email : [hakivinamana@student.uns.ac.id](mailto:hakivinamana@student.uns.ac.id)

[milarosyidauswatunnisa@student.uns.ac.id](mailto:milarosyidauswatunnisa@student.uns.ac.id)

[wafirotullaila@student.uns.ac.id](mailto:wafirotullaila@student.uns.ac.id)

Asisten : Haidar Hendri Setyawan

Taufiq Odhi Dwi Putra

*Kata kunci*: Multi-thread, Socket , Single-thread

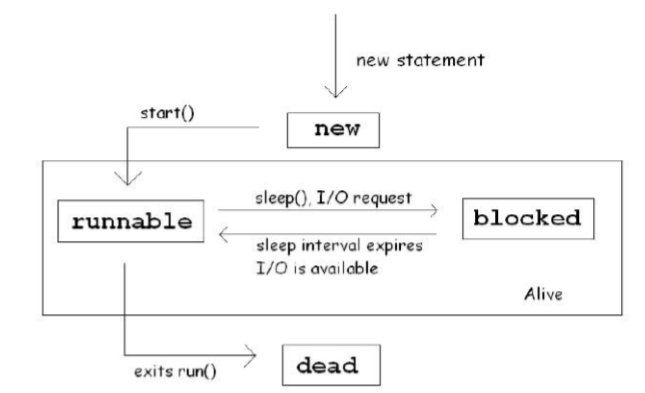
# Pendahuluan

Suatu proses dikontrol oleh paling sedikit satu thread. Namun, sebagian besar proses yang ada sekarang biasanya dijalankan oleh beberapa buah thread. Multithreading adalah sebuah mekanisme di mana dalam suatu proses, ada beberapa thread yang mengerjakan tugasnya masing-masing pada waktu yang bersamaan. Contohnya, sebuah web browser harus menampilkan sebuah halaman yang memuat banyak gambar. Pada program yang single-threaded, hanya ada satu thread untuk mengatur suatu gambar, lalu jika gambar itu telah ditampilkan, barulah gambar lain bisa diproses. Dengan multithreading, proses bisa dilakukan lebih cepat jika ada thread yang menampilkan gambar pertama, lalu thread lain untuk menampilkan gambar kedua, dan seterusnya, di mana thread-thread tersebut berjalan secara paralel.

# Dasar Teori

* 1. *Pemrograman Thread*

Saat sebuah program Java dieksekusi, yaitu saat main() dijalankan, ada sebuah thread utama yang bekerja. Java adalah bahasa pemrograman yang mendukung adanya pembentukan thread tambahan selain thread utama tersebut. Thread dalam Java diatur oleh Java Virtual Machine (JVM) sehingga sulit untuk menentukan apakah thread Java berada di user-level atau kernel-level. Diagram state pemrograman multithread pada java dapat ditunjukkan sebagai berikut :



Gambar 1. Diagram State Multi-Threading pada Java

Suatu thread bisa berada pada salah satu dari status berikut:

1. New

Thread yang berada di status ini adalah objek dari kelas Thread yang baru dibuat, yaitu saat instansiasi objek dengan statement new. Saat thread berada di status new, belum ada sumber daya yang dialokasikan, sehingga thread belum bisa menjalankan perintah apapun.

1. Runnable

Agar thread bisa menjalankan tugasnya, method start() dari kelas Thread harus dipanggil. Ada dua hal yang terjadi saat pemanggilan method start(), yaitu alokasi memori untuk thread yang dibuat dan pemanggilan method run(). Saat method run() dipanggil, status thread berubah menjadi runnable, artinya thread tersebut sudah memenuhi syarat untuk dijalankan oleh JVM. Thread yang sedang berjalan juga berada di status runnable.

1. Blocked

Sebuah thread dikatakan berstatus blocked atau terhalang jika terjadi blocking statement, misalnya pemanggilan method sleep(). sleep() adalah suatu method yang menerima argumen bertipe integer dalam bentuk milisekon. Argumen tersebut menunjukkan seberapa lama thread akan "tidur". Selain sleep(), dulunya dikenal method suspend(), tetapi sudah disarankan untuk tidak digunakan lagi karena mengakibatkan terjadinya deadlock. Di samping blocking statement, adanya interupsi M/K juga dapat menyebabkan thread menjadi blocked. Thread akan menjadi runnable kembali jika interval method sleep()-nya sudah berakhir, atau pemanggilan method resume() jika untuk menghalangi thread tadi digunakan method suspend() atau M/K sudah tersedia lagi.

1. Dead

Sebuah thread berada di status dead bila telah keluar dari method run(). Hal ini bisa terjadi karena thread tersebut memang telah menyelesaikan pekerjaannya di method run(), maupun karena adanya pembatalan thread. Status jelas dari sebuah thread tidak dapat diketahui, tetapi method isAlive() mengembalikan nilai boolean untuk mengetahui apakah thread tersebut dead atau tidak.

* 1. *Java Thread*

Sebuah thread mulai dibuat, ketika dilakukan pembuatan objek baru dari class java.lang.Thread. Sebuah objek Thread mewakili thread sesungguhnya dalam interpreter java dan juga memberikan penangan control dan sinkronisasi eksekusinya. Sebuah objek yang diperlakukan sebagai sebuah thread harus menerapksi dari sebuah interface java.lang.runnable. Dalam interface ini menyediakan suatu method kunci run().

public interface Runnable{

Abstract public void run();

}

Sebuah thread dimulai siklus hidupnya dengan memanggil method run(). Untuk membuat sebuah class yang akan bertindak sebagai sebuah thread, didefinisikan dengan bentuk sebagai berikut :

Class namaClass implements Runnable {

Void run(){

//implementasi kode..... }

}

Sedangkan cara pemanggilannya dengan :

* Buat instance objek dari class yang mengimplemntasikan interface Runnable tersebut.
* Buat object dari class java.lang.Thread dengan parameter konstruktor object dari class yang mengimplemnetasikan interface Runnable
* Panggil method start() dari objek yang mengimplementasikan interface Runnable.

Cara lain untuk membuat thread adalah dengan membuat class turunan dari class lain yang telah menerapkan Runnable. Class Thread merupakan class yang mengimplementasikan interface Runnable. Berikut contoh :

class namaClass extends Thread {

public void run(){

//kode ............... }

}

* 1. *Pengontrolan Thread*

Method start() digunakan untuk memulai sebuah thread dieksekusi. Method stop() digunakan untuk menghentikan eksekusi suatu thread. Method suspend() digunakan untuk menghentikan sementara waktu (pause) untuk kemudian dapat dilanjutkan kembali dengan menggunakan method resume(). Jika beban yang harus dikerjakan oleh thread berat/mahal, maka dapat digunakan method suspend() dan resume(). Method static sleep(n\_msec) digunakan untuk menghentikan thread yang active selama n millisecond*.*

try {

Thread.sleep(1000);

} catch(Exception e ){ }

Dalam pembuatan aplikasi nantinya kita mungkin akan menggunakan beberapa thread untuk mengakses resource yang sama. Untuk menghindari masalah pada satu pengaksesan resource yang sama antar thread, maka satu thread harus tahu akan thread yang lain. Inilah yang disebut dengan synchronization. Struktur penggunaan synchronization adalah

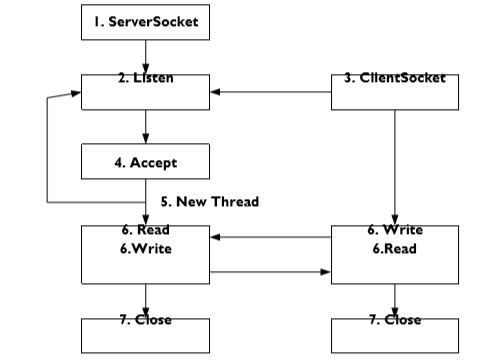
[public] synchronized [type\_data] namaMethod (parameter masukkan){

//koding......

}

* 1. *Penggunaan Mutlthreading dalam Pemrograman Socket*

Penggunaan multithreading untuk pemrograman socket sangat penting sekali, karena suatu server harus melayani banyak client. Seperti pada modul yang pertama, aplikasi socket untuk server hanya dapat melayani client tunggal, setalah itu server berhenti. Dengan menggunakan konsep pemrograman multithreading, maka setiap client koneksi ker server akan diciptakan suatu thread baru, demikian seterusnya untuk client selanjutnya. Sehingga server selalu running dan siap menerima request dari client (seperti program web server apache). Jika kita gambarakan dalam diagram blok sebagai berikut

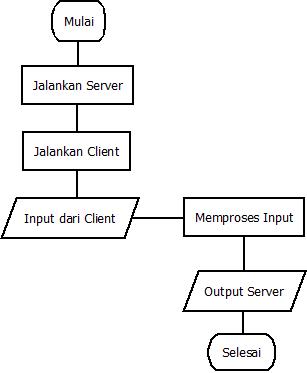


Gambar 2. Diagram Blok Pemrograman Socket

# ALAT dan LANGKAH PERCOBAAN

1. Laptop/PC
2. IDE Java

Langkah percobaan:



Gambar 3. Langkah percobaan

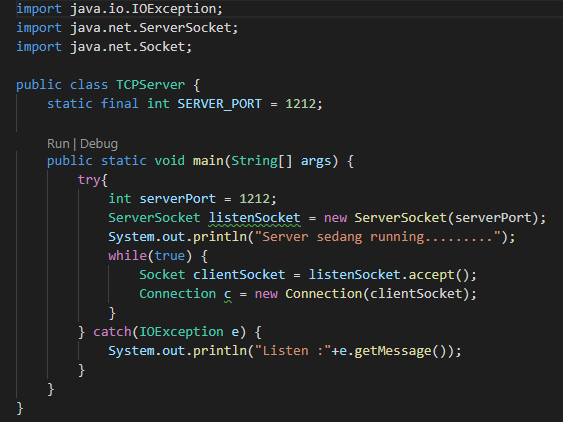
# Hasil dan Analisis Percobaan

* 1. Soal Latihan
* Buatlah socket programming client server (TCP) dengan konsep multi-threading untuk proses penjumlahan, maksudnya adalah buat suatu socket server yang didalamnya terdapat suatu method untuk melakukan penjumlahan suatu data array bertype integer (ambil indeks maksimal 5)x, pada socket client mengirimkan data yang akan di jumlahkan, dan socket server malakukan proses penjumlahan dan mengirimkan hasilnya ke socket client.
* Buatlah untuk kasus yang sama dengan paket Datagram (UDP).
  1. Hasil dan Analisis Percobaan
     1. Pembahasan 1

Pada soal 1 adalah membangun pemrograman socket TCP dengan multi-threading untuk proses penjumlahan. Pemrograman ini terdiri atas 3 file .java yaitu :

* *TCPServer.java*

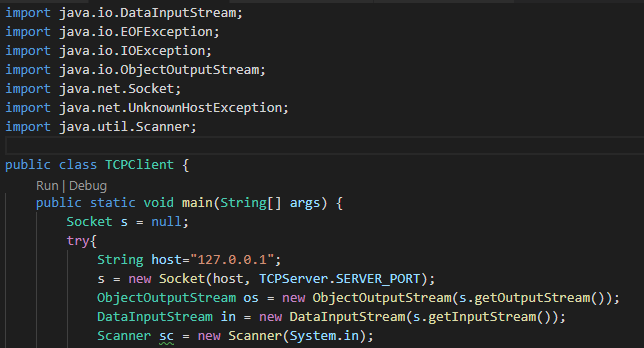
Dalam *TCPServe*r.java berisikan *source code* untuk membangun socket server dengan model TCP. *Source code*  ini merupakan proses pertama pada *socket programming* yaitu *open socket*. Pada *TCPServe*r.java ini , server bertanggung jawab saat client melakukan request data ke server.



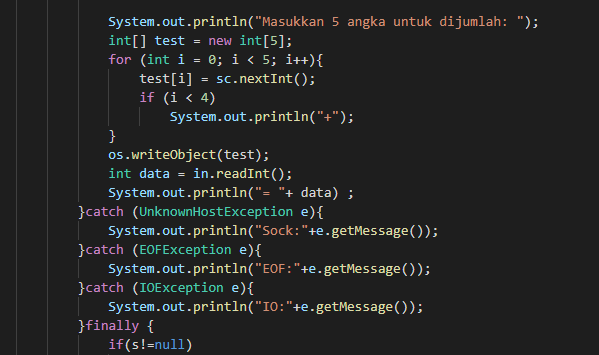
Gambar 4. *Source code TCPServe*r.java

* *TCPClient.java*

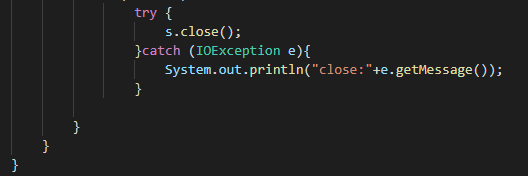
Pada *TCPClient.java* ini terdapat beberapa proses antara lain, *open socket, input output stream,* dan *close socket.* Pada proses *open socket ,* client meminta request kepada server. Kemudian setelah server menerima, maka server akan merespon dan menyuruh client untuk melakukan input output data. Pada proses input output data, client akan memberikan request data yang akan diberikan ke server. Oleh karena pada program kali ini adalah proses penjumlahan, maka client harus meng-inputkan data sebanyak 5 data dalam bentuk tipe data integer.



Gambar 5. *Open socket programming sisi client*



Gambar 6. *Input output data* sisi client



Gambar 7. *Close socket sisi client*

* *Connection.java*

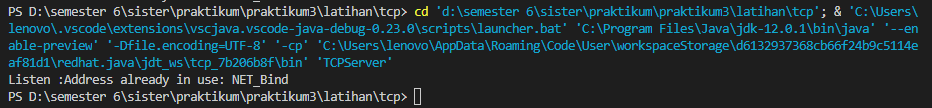
Dalam melakukan request data, client diminta melakukan input data sejumlah 5 data integer yang akan dilakukan proses penjumlahan. Oleh karena program dibangun dengan konsep multi-threading , maka hasil penjumlahan 5 data tersebut bisa diproses sekali.



Gambar 8. *Source code socket TCP multi-thread*

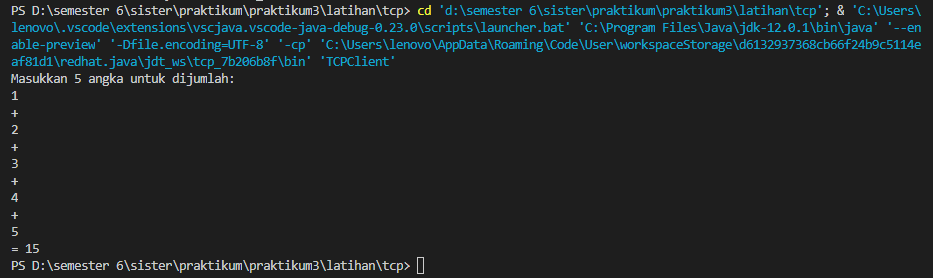
* Hasil Running Percobaan

Untuk menjalankan proses penjumlahan tersebut , yang dilakukan pertama adalah melakukan eksekusi atau *compile* pada tcpServer.java



Gambar 9. Hasil running *TCPServer.java*

Gambar diatas adalah hasil saat tcpServer.java dijalankan. Hal ini berarti bahwa server siap untuk menerima input yang akan diberikan oleh client. Kemudian setelah server siap, kita menjalankan dari sisi client yang eksekusi tcpClient.java. Pada sisi client , dapat memberikan input yang akan diterima oleh server. Input berupa angka karena dalam hal ini merupakan proses penjumlahan. Setelah server menerima 5 angka yang diinputkan client, maka server akan melakukan proses penjumlahan dan akan memberikan hasil penjumlahan.



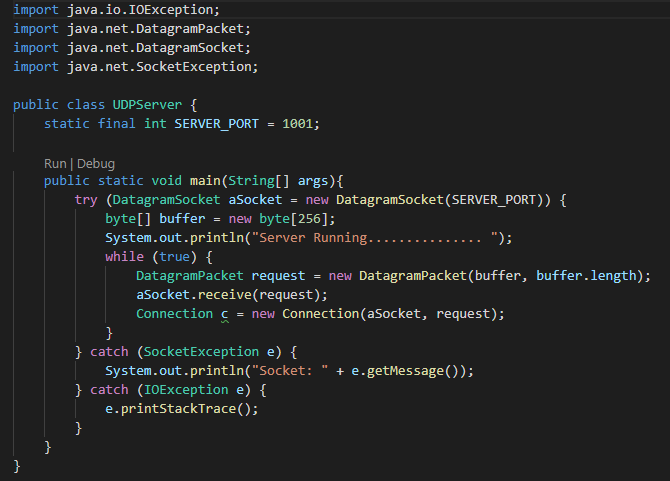
Gambar 10. Hasil running *TCPClient.java*

* + 1. Pembahasan 2

Pada soal 2 adalah membangun pemrograman socket UDP dengan multi-threading untuk proses penjumlahan. Pemrograman ini terdiri atas 3 file .java yaitu :

* *UDPServer.java*

*UDPServer.java* berisikan *source code*  untuk membangun socket server dengan model UDP.

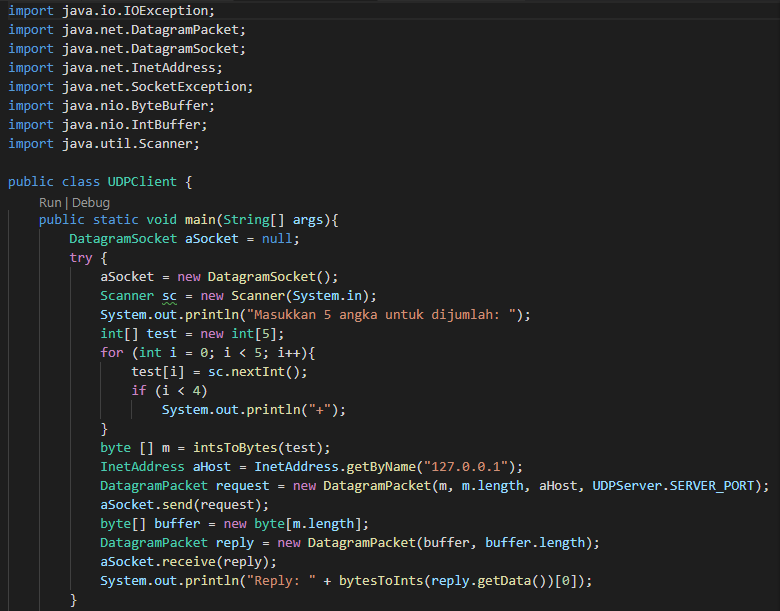


Gambar 10. *Source code socket server with UDP*

Di atas adalah *source code* untuk membangun socket server dengan model UDP. Pada program ini, awalnya server yang berupa datagram akan berjalan dan menunggu adanya input. Input ini akan diterima dan selanjutnya ditampilkan. Di dalam program server ini sendiri terdapat fungsi penjumlahan yang akan melakukan proses penjumlahan pada input yang diterima oleh program. Kemudian hasil dari fungsi ini akan menjadi hasil akhir program.

* *UDPClient.java*

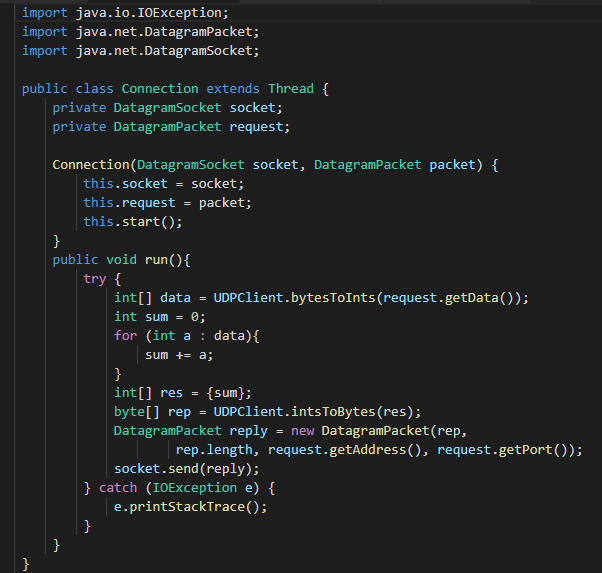
*UDPClient.java* berisikan *source code* untuk membangun socket client dengan model UDP. Seperti halnya pada program server, program client ini akan berjalan dan kemudian menerima input. Input ini kemudian dikirimkan ke program server dan dilakukan pengolahan sesuai dengan fungsi yang ada pada program server.



Gambar 11. *Source code socket client with UDP*

* *Connection.java*

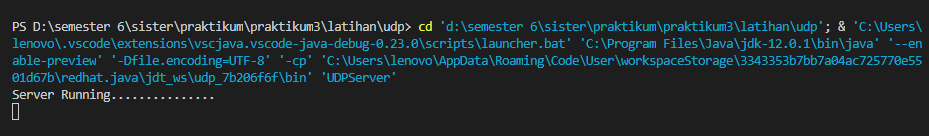
Dalam melakukan request data, client diminta melakukan input data sejumlah 5 data integer yang akan dilakukan proses penjumlahan. Oleh karena program dibangun dengan konsep multi-threading , maka hasil penjumlahan 5 data tersebut bisa diproses sekali.



Gambar 12. *Source code socket UDP multi-thread*

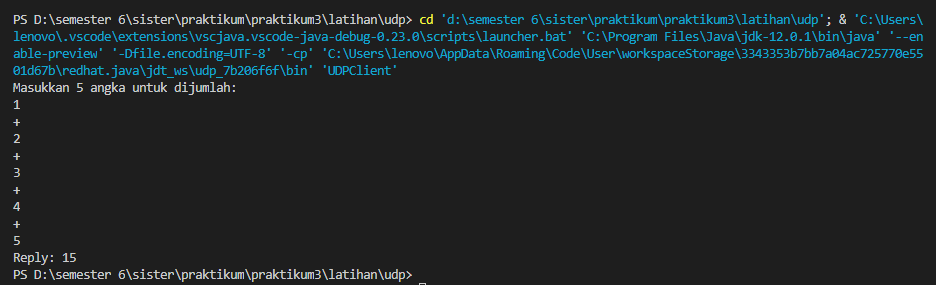
* Hasil Running Percobaan

Untuk menjalankan proses penjumlahan tersebut , yang dilakukan pertama adalah melakukan eksekusi atau *compile* pada *UDPServer.java*



Gambar 13. Hasil running *UDPServer.java*

Gambar diatas adalah hasil saat UDPServer.java dijalankan. Hal ini berarti bahwa server siap untuk menerima input yang akan diberikan oleh client. Kemudian setelah server siap, kita menjalankan dari sisi client yang eksekusi UDPClient.java. Pada sisi client , dapat memberikan input yang akan diterima oleh server. Input berupa angka karena dalam hal ini merupakan proses penjumlahan. Setelah server menerima 5 angka yang diinputkan client, maka server akan melakukan proses penjumlahan dan akan memberikan hasil penjumlahan.



Gambar 14. Hasil running *UDPClient.java*

# IV.Kesimpulan

Berdasarkan hasil dan analisis diatas dapat disimpulkan bahwa :

1. Multi-threading dapat diterapkan dalam pemrograman socket baik dengan TCP dan UDP.
2. Dengan menerapkan Multi-threading, server dapat menerima banyak client.

# DaftarPustaka

1. Modul Praktikum Pemrograman Socket Dengan Konsep Multi-Threading

**[Hakikid Amana].** Mahasiswa Informatika angkatan 2017.

**[Mila Rosyida Uswatunnisa].** Mahasiswa Informatika

**[Wafirotul Laila].** Mahasiswa Informatika angkatan 2017.