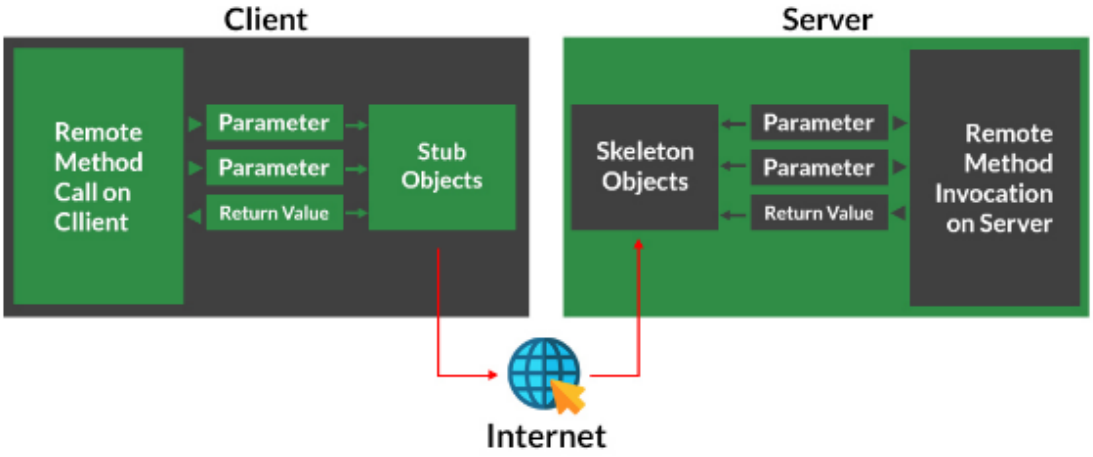
MODUL 6
Remote Method Invocation (RMI)

A. Tujuan :

1. Memahami tentang konsep RMI
2. Mengaplikasikan RMI dalam interkoneksi client-server

B. Dasar Teori

RMI (*Remote Method Invocation*) adalah sebuah API yang menyediakan mekanisme untuk membuat aplikasi terdistribusi di Java. RMI memungkinkan sebuah object untuk memanggil metode pada object yang berjalan di JVM lain.

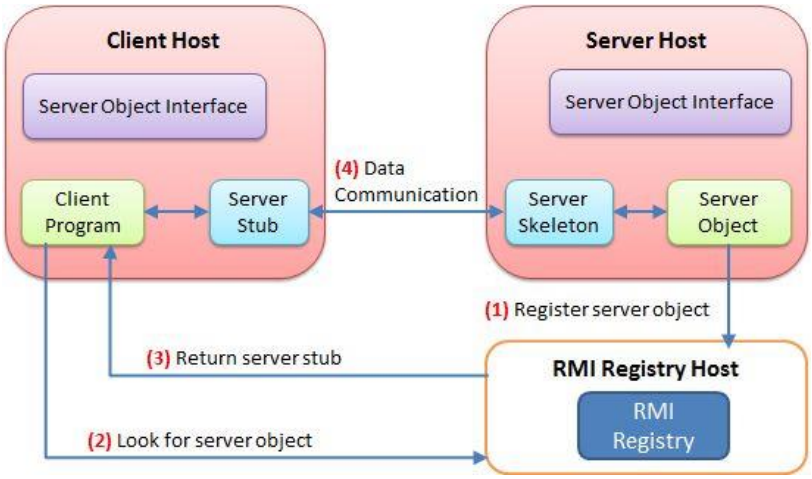


Gambar 1. Permohonan akses method pada *remote object* dan eksekusi pada *remote machine*

Sistem yang menggunakan RMI untuk komunikasi biasanya dibagi menjadi dua kategori: klien dan server. Sebuah server menyediakan layanan RMI, dan klien memanggil metode object layanan ini.
Server RMI harus mendaftar pada layanan pencarian (lookup service) sehingga klien dapat menemukan mereka. Dalam platform Java terdapat sebuah aplikasi yang disebut *rmiregistry*, yang berjalan sebagai proses terpisah dan memungkinkan aplikasi untuk mendaftar layanan RMI atau mendapatkan referensi ke nama layanan tertentu. Setelah server telah terdaftar, maka ia akan menunggu permintaan RMI masuk dari klien.

Client RMI akan mengirimkan pesan RMI untuk meminta object method dari jarak jauh. Untuk melakukan permintaan, client harus memiliki referensi object remote yang bisa diperoleh dengan mencari layanan pada *RMI registry*. Aplikasi client meminta nama layanan tertentu dan menerima URL. Format berikut digunakan untuk merepresentasikan sebuah referensi *remote object*.
rmi://hostname:port/servicename
dimana hostname merepresentasikan nama server (atau ip address), port lokasi layanan pada mesin dan servicename sebuah string yang mendiskripsikan layanan.

RMI menyediakan komunikasi jarak jauh antara aplikasi menggunakan dua object *stub* dan *skeleton*. RMI menggunakan object stub dan object skeleton untuk komunikasi dengan object remote. Sebuah remote object adalah obyek yang metodenya dapat dipanggil dari JVM lain.
Stub adalah obyek yang bertindak sebagai gateway untuk sisi klien. Semua permintaan keluar disalurkan melalui object ini. Sedangkan *Skeleton* adalah obyek, bertindak sebagai gateway untuk object sisi server. Semua permintaan masuk akan dialihkan melalui object ini.



Gambar 2. Stub pada client RMI memanggil skeleton pada server RMI

Implementasi Layanan RMI
Suatu sistem yang menggunakan RMI harus menggunakan sebuah layanan interface. Layanan interface mendefinisikan method object yang dapat dikontrol dari jarak jauh, parameter tertentu, mengembalikan (*return*) tipe, dan *exception*. Object stub dan skeleton, begitu juga dengan layanan RMI diimplementasikan pada interface ini.
Berikut ini implementasi layanan RMI dengan Java, yang terdiri dari: membuat object RMI, membuat server RMI, dan membuat client RMI.

C. Tugas Pendahuluan
Buatlah desain flowchart untuk setiap soal dalam percobaan

D. Percobaan
Sebelum membuat pemrograman java dengan RMI, pastikan bahwa versi java yang digunakan adalah JDK 60 dan harus sama baik di **java** maupun **javac** seperti pada gambar berikut ini.

```
D:\PENS\DATA KULIAH\Sistem Terdistribusi\RMI\p2>java -version
java version "1.8.0_60"
Java(TM) SE Runtime Environment (build 1.8.0_60-b27)
Java HotSpot(TM) 64-Bit Server VM (build 25.60-b23, mixed mode)

D:\PENS\DATA KULIAH\Sistem Terdistribusi\RMI\p2>javac -version
javac 1.8.0_60
```

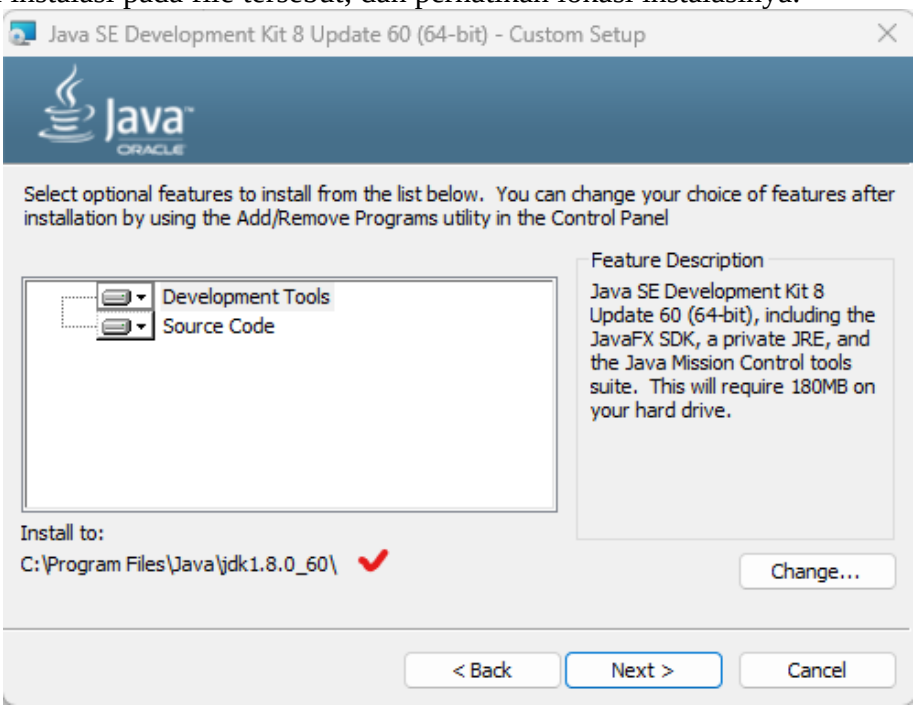
Hal ini dikarenakan, RMI sudah tidak didukung di java versi terbaru.

Untuk software JDK versi 60 adalah sebagai berikut:
Java SE Development Kit 8 Update 60 (64-bit) jdk-8u65-windows-x64.exe x64

Dan bisa didownload disini:
https://drive.google.com/drive/folders/1_P2DPUe2NUMTHBGm0cyp4LIuj52oV1jx?usp=share_link

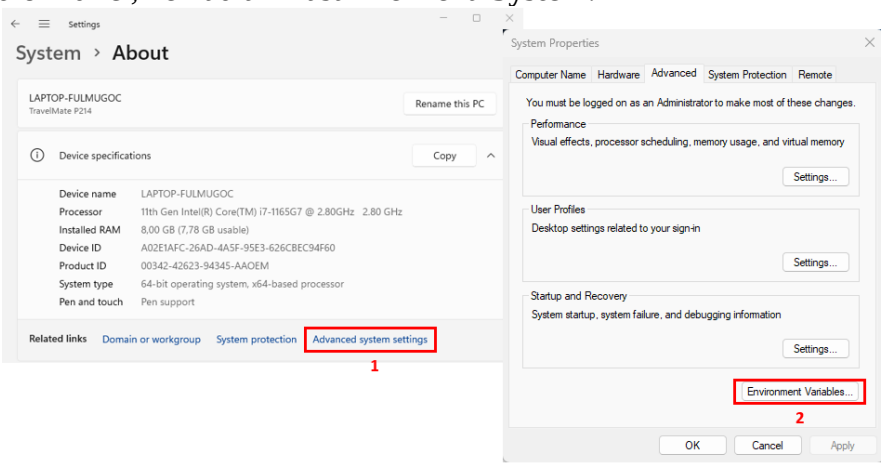
jdk-8u60-windows-x64

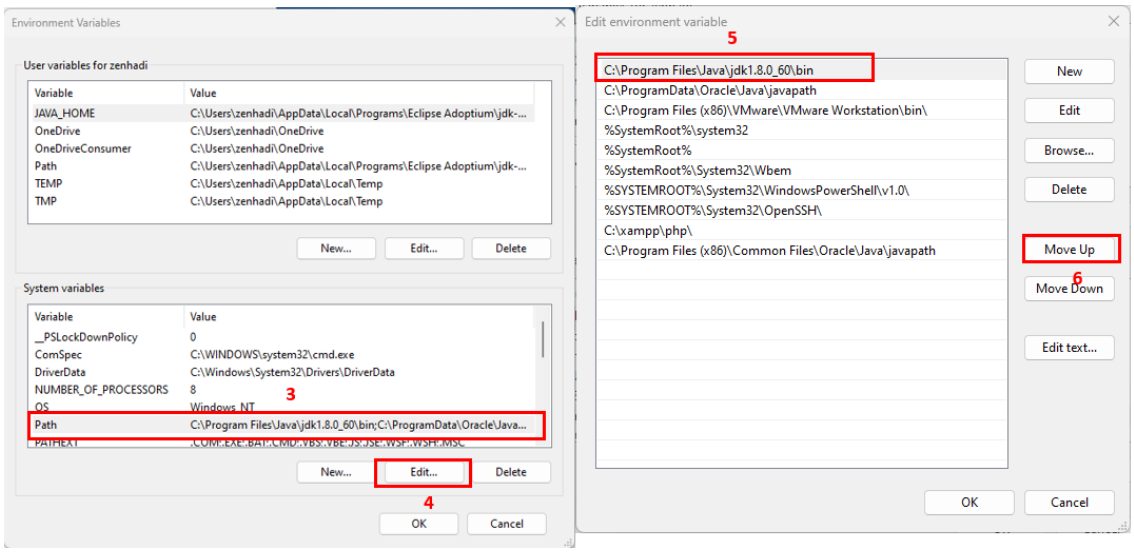
Lakukan instalasi pada file tersebut, dan perhatikan lokasi instalasinya.



Jika sudah selesai proses instalasinya selanjutnya set ke path windows agar pada CMD akan terintegrasi Java SE yang nantinya akan digunakan untuk menjalankan method dan mengcompilenya.

Dari **Control Panel**, kemudian masuk ke menu **System**.





Pastikan pada langkah 5 bahwa **JDK versi 60** ada di posisi paling atas agar dijalankan terlebih dahulu ketika membuka CMD.
Setelah itu lakukan restart komputer dan pastikan **versi** pada **java** dan **javac** sudah sama.

D.1. Latihan

1. Membuat aplikasi HelloWorld dengan RMI

Dalam hal ini, akan dibuat method RMIServer dan RMIClient, dan diperlukan 4 buah file untuk mengaplikasikan RMI.

- HelloWorld.java: untuk membuat obyek remote
- HelloImpl.java: untuk membuat Remote Implements
- HelloServer: server yang akan dijalankan
- HelloClient: client yang akan mengakses layanan server

Buatlah folder untuk menyimpan semua file diatas.

a. Membuat object RMI: HelloWorld.java

Buka file notepad dan simpan dengan nama **HelloWorld.java**

```
//defining remote interface

import java.rmi.*;

public interface HelloWorld extends Remote {
        public String display() throws RemoteException;
}
```

b. Membuat implement object RMI: HelloImpl.java

Buka file notepad dan simpan dengan nama **HelloImpl.java**

```
//implementation of remote interface
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;

public class HelloImpl extends UnicastRemoteObject implements HelloWorld {
        public HelloImpl() throws RemoteException {}

        public String display() throws RemoteException {
                return ("Hello world from server");
        }
}
```

c. Membuat Server RMI: HelloServer.java

Buka file notepad dan simpan dengan nama **HelloServer.java**

Server digunakan sebagai tempat untuk melakukan sharing object, sehingga client dapat mengakses object RMI melalui servre tersebut. Dalam RMI, server direpresentasikan sebagai interface `java.rmi.registry.Registry`. Karena merupakan interface, maka untuk membuat object `java.rmi.registry.registry` dapat dilakukan dengan instansiasi "new".

Method `bind()` memiliki dua parameter, pertama adalah String dan yang kedua adalah Object. String tersebut adalah nama object yang disimpan, nama object tersebut bebas sesuai yang diinginkan namun nantinya harus sama dengan string nama object pada client agar bisa diakses. Sedangkan parameter kedua, yaitu object yang dishare.

Perlu diperhatikan pula pada penggunakan method `bind()` adalah jika dilakukan penyimpanan object dengan nama yang telah ada di server, maka akan terjadi error. Misal dilakukan penyimpanan terhadap object "data" kemudian menyimpan object baru dengan maksud memperbarui object sebelumnya dengan nama yang sama, maka ini akan menyebabkan error. Oleh karena itu, untuk penyimpanan object yang sewaktu-waktu bisa berubah maka disarankan menggunakan method `rebind()`.

Cara kerja method `rebind()` adalah pertama server akan mendeteksi apakah object dengan nama yang sama telah ada, jika belum maka object yang baru akan disimpan, namun jika object dengan nama yang sama telah ada maka object yang lama akan dihapus dan digantikan oleh object yang baru.

```
//SERVER PROGRAM

import java.rmi.*;
import java.net.*;
import java.io.*;

public class HelloServer {
        public static void main (String args[]) {
                try {
                        HelloImpl localobj = new HelloImpl();
                        Naming.rebind ("rmi://localhost/HelloWorld",localobj);
                } catch (RemoteException re) {
                        re.printStackTrace();
                }
                catch (MalformedURLException mfe) {
                        mfe.printStackTrace();
                }
        }
}
```

d. Membuat Client RMI: HelloClient.java

Buka file notepad dan simpan dengan nama **HelloClient.java**

Client sebagai pihak yang mengakses object yang ada pada server. Untuk membuat client dilakukan dengan menggunakan class java.rmi.Naming. Naming merupakan class utilities, artinya tidak bisa membuat object Naming dengan instansiasi "new", sehingga cukup mengetikkan java.rmi.Naming untuk membuat client.
Selanjutnya untuk mengakses Object pada server maka harus membuat object interfacenya, bukan classnya. Misal pada contoh pengenalan RMI ini dengan membuat object interface DataInterface yang merupakan interface yang diimplementasikan oleh class Data. Untuk membuat object interface tersebut dilakukan dengan:

```
HelloWorld remobject = (HelloWorld) Naming.lookup("rmi://"+host+"/HelloWorld");
```

Untuk mengakses data tersebut dapat menggunakan method lookup yang dimiliki oleh class Naming dengan parameter url yang sama dengan yang didefinisikan pada server. Berikut ini format url pengaksesan object di server:

```
"rmi://"+host+"/"+nama_object
```

Dimana host adalah tempat server, baik berupa nama host maupun IP Address, untuk port adalah port yang digunakan oleh server dan untuk nama_object adalah nama object yang ada di server.

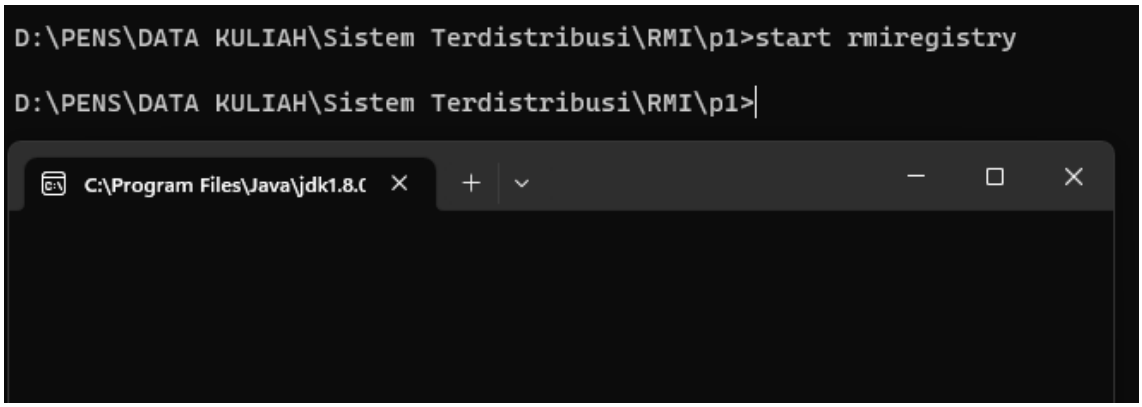
```
//CLIENT PROGRAM
import java.rmi.*;
import java.net.*;
import java.io.*;

public class HelloClient {
        public static void main (String args[]) {
                try {
                        String host="localhost";
                        HelloWorld remobject =
(HelloWorld)Naming.lookup("rmi://"+host+"/HelloWorld");
                        System.out.println (remobject.display());
                } catch (RemoteException re) {
                        re.printStackTrace();
                }
                catch (NotBoundException nbe) {
                        nbe.printStackTrace();
                }
                catch (MalformedURLException mfe) {
                        mfe.printStackTrace();
                }
        }
}
```

e. Jalankan di sisi server

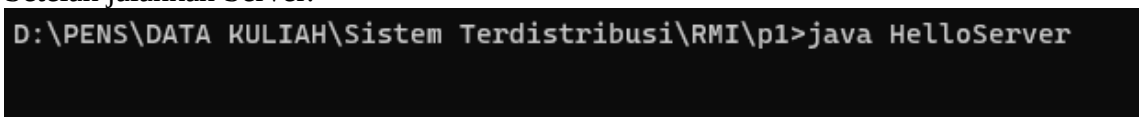
Buka aplikasi CMD dan kompilasi file yang diperlukan oleh server.

f. Jalankan RMI Registry

Ketika menjalankan RMI Registry, maka akan terbuka CMD yang baru dan kosong. Biarkan dan jangan ditutup karena itu sebagai layanan yang diberikan oleh server.

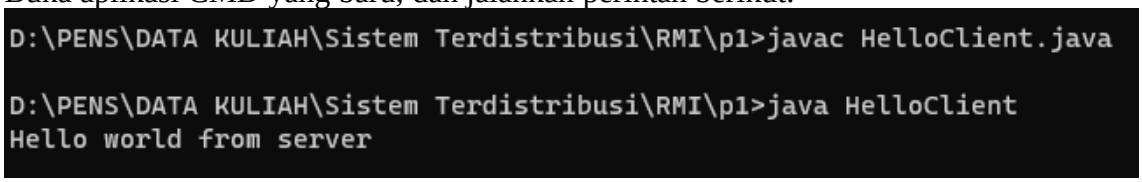


Setelah jalankan Server.



g. Jalankan di sisi client

Buka aplikasi CMD yang baru, dan jalankan perintah berikut.



Hasilnya terdapat kalimat yang dikirim oleh server ke client.

2. Membuat aplikasi penjumlahan

Aplikasi berikut adalah untuk menjumlahkan 2 bilangan yang dikirim oleh Client. Kemudian Server akan melakukan perhitungan dan mengembalikan hasilnya ke Client. Terdapat 4 file yang harus dibuat:

- AddRem.java: untuk membuat obyek remote
- AddRemImpl.java: untuk membuat Remote Implements
- AddServer: server yang akan dijalankan
- AddClient: client yang akan mengakses layanan server

a. File AddRem.java

```
//define remote interface

import java.rmi.*;

public interface AddRem extends Remote {
        public int addNum (int a, int b) throws RemoteException;
}
```

b. File AddRemImpl.java

```
//implementation of interface

import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;

public class AddRemImpl extends UnicastRemoteObject implements AddRem {
        public AddRemImpl() throws RemoteException{}
        public int addNum(int a, int b) {
                return (a+b);
        }
}
```

c. File AddRemImpl.java

```
//implementation of interface

import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;

public class AddRemImpl extends UnicastRemoteObject implements AddRem {
        public AddRemImpl() throws RemoteException{}
        public int addNum(int a, int b) {
                return (a+b);
        }
}
```

d. File AddServer.java

```
//Server Program

import java.rmi.*;
import java.net.*;

public class AddServer {
        public static void main (String args[]) {
                System.out.println ("Server sedang menunggu koneksi");
                try {
                        AddRemImpl locobj = new AddRemImpl();
                        Naming.rebind ("rmi://localhost/AddRem",locobj);
                } catch (RemoteException re) {
                        re.printStackTrace();
                } catch (MalformedURLException mfe) {
                        mfe.printStackTrace();
                }
        }
}
```

e. File AddClient.java

```
//Client Program

import java.rmi.*;
import java.net.*;
import java.io.*;
import java.util.*;

public class AddClient {
        public static void main (String args[]) {
                String host="localhost";
                Scanner sc = new Scanner (System.in);
                System.out.println ("Masukkan nilai ke-1 : ");
                int a = sc.nextInt();
                System.out.println ("Masukkan nilai ke-2 : ");
                int b = sc.nextInt();
                try {
                        AddRem remobj = (AddRem)Naming.lookup("rmi://"+host+"/AddRem");
                        System.out.println (remobj.addNum(a,b));
                } catch (RemoteException re) {
                        re.printStackTrace();
                } catch (NotBoundException nbe) {
                        nbe.printStackTrace();
                } catch (MalformedURLException mfe) {
                        mfe.printStackTrace();
                }
        }
}
```

Lakukan langkah-langkah seperti sebelumnya:

a. Kompilasi obyek dan implement-nya
b. Kompilasi Server dan jalankan
c. Jalankan RMI Registry
d. Kompilasi Client dan jalankan
e. Amati hasilnya dan catat

D.2. Permasalahan

1. Buatlah program ***salam*** dengan RMI. Ketika client memasukkan namanya, kemudian server menjawab: "Assalamu 'alaikum [nama_client]"
   Seperti contoh berikut:

```
D:\PENS\DATA KULIAH\Sistem Terdistribusi\RMI\p5>java AddClient
Masukkan nama : zenhadi
Jawaban Server : Assalamu 'alaikum zenhadi
```

2. Buatlah miniatur DNS (Doman Name Services) dengan menggunakan RMI.
   a. Buat file dns.txt yang isinya sebagai berikut:

```
100.10.20.11      www.rmi1.com
150.10.20.11      www.rmi2.com
200.10.20.11      www.rmi3.com
```

   b. Modifikasi program latihan 2, bila client mengirim alamat website maka server akan memberikan informasi alamat IP seperti contoh berikut:

```
D:\PENS\DATA KULIAH\Sistem Terdistribusi\RMI\p4>java AddClient
Masukkan alamat web : www.rmi2.com
Alamat IP adalah : 150.10.20.11 www.rmi2.com
```

Petunjuk pada sisi server:

a. Gunakan operasi file di java untuk membaca file text seperti contoh di link berikut: https://www.w3schools.com/java/java_files_read.asp
b. Lakukan pembacaan per baris perintah pada file dns.txt, bila ditemukan maka hasilnya akan di-return ke client.

E. Laporan Resmi :

1. Analisalah semua program diatas dan buat kesimpulan.