# Modul Praktikum

# Sistem Terdistribusi

Pengampu
Winarno, S.Si., M.Eng

**Informatika**
**Universitas Sebelas Maret**
**Surakarta**

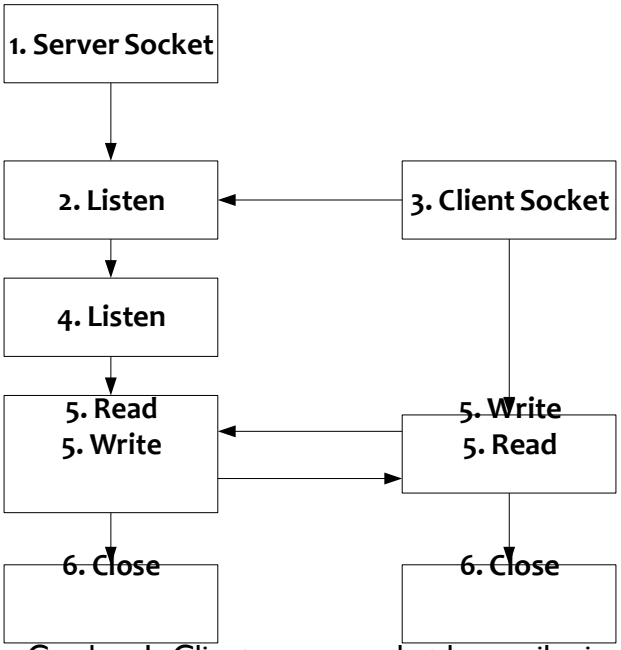
# MODUL I
# PEMROGRAMAN SOCKET DENGAN PAKET TCP
***(Menggunakan Pemrograman Java)***

## A. Tujuan

1. Menjelaskan mengenai kosep komunikasi client-server dengan socket
2. Menjelaskan pembuatan aplikasi socket dengan paket TCP

## B. Pendahuluan

Socket adalah sebuah abstraksi perangkat lunak yang digunakan sebagai suatu "terminal" dari suatu hubungan antara dua mesin atau proses yang saling berinterkoneksi. Socket termasuk model komunikasi client-server yang dapat digambarkan proses komunkasinya sebagai berikut

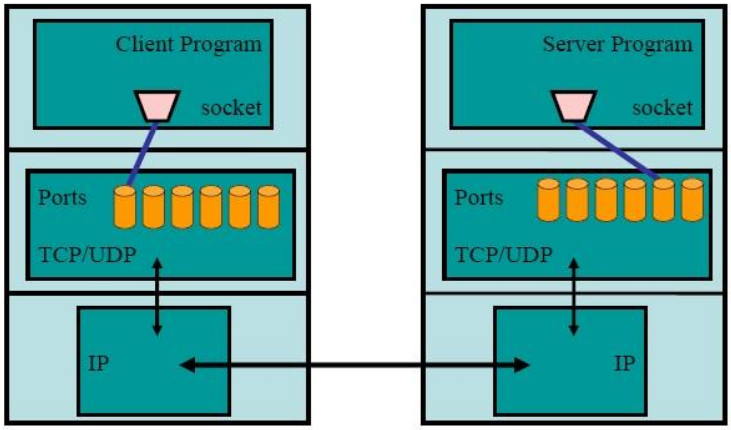



Gambar 1. Client-server socket komunikasi

Berdasarkan gambar tersebut dapat kit abaca bahwa socket dapat melakukan beberapa operasi yang meliputi koneksi ke mesin remote, mengirim data (write), menerima data (read), menutup koneksi (close), bind to port, listen pada data yang masuk dan menerima koneksi dari mesin remote pada port tertentu. Dalam socket programming, tiap mesin yang saling berinteraksi harus terpasang socket pula.

Dalam socket programming dapat dibangun dengan menggunakan dua model paket TCP (untuk connection oriented) dan UDP (untuk connectionless). Jika

digambarkan komunikasi socket server dan socket client dengan jenis paketnya sebagai berikut




Gambar 2. Socket to socket communication

## Data Stream

Inti dalam input/output dalam pemrograman java adalah data stream. Data stream adalah aliran data yang berupa aliran byte atau karakter dari devaice input ke devaice output pada saat program dieksekusi. Berfikir tentang sebuah data stream adalah berfikir tentang sebuah *pipeline*. Kita dapat menulis informasi pada *pipeline* (write), atau kita juga dapat memperoleh informasi/mengambil informasi dari *pipeline* (read).

Dengan socket, suatu informasi hanya dapat dioperasikan dengan dengan cara read dan write. Kita dapat menulis informasi dengan menggunakan class *OutputStream* (dimana dalam *OutputStream* menyediakan method write()), dan kita dapat membaca/mengambil informasi dengan menggunakan class InputStream (dimana didalamnya disediakan method red()). Ketika sebuah koneksi sukses, setiap end-point akan menciptakan InputStream dan OutputStream. InputStream di ciptakan dengan menggunakan method *getInputStream()*, dan *OutputStream* diciptakan dengan menggunakan method *getOutputStream()*. Dalam stream ini membolehkan kita untuk melakukan proses read dan write baik single atau multiple byte. Beberapa class lain yang dapat digunakan dalam socket programming yang terbendel dalam package java.io.* adalah :

- *BufferedReader* dan *BufferedWriter*, keduanya untuk input dan output stream dengan menggunakan buffer. Dengan menggunakan buffer akan lebih efisien jika dibandingkan yang tanpa menggunakan buffer.
- *FilterInputStream* dan *FilterOutputStream*, keduanya dapat dibangun dengan membuat instace dari InputStream dan OutputStream (keduanya tidak menggunakan buffer). Perbaikan kinerja yang diberikan adalah dengan adanya konsep caching dan flushing.

- *DataInputStream* dan *DataOutputStream* menyediakan servis tingkat tinggi untuk proses read dan write terhadap type data primitive (ingat kembali kuliah OOP).
- *FileReader*, *FileWriter*, *FileInputStream* dan FileOutputStream, semuanya digunakan untuk pembacaan dan penulisan dari dan ke suatu file.

**TCP Socket**

Karakteristik dari TCP adalah connection oriented, reliable dan statefule. Connection oriented maksudnya adalah bahwa dalam komunikasinya TCP sebelum proses komunikasi terlebih dahulu dilakukan perjanjian/kesepakatan, baru kemudian terjadi komunikasi. Reliable, artinya bahwa paket TCP akan dikirimkan sampai ke tujuan, jika ada masalah dalam perjalanannya, maka dilakukan retransmisi. Sedangkan untuk stetaful, karena dalam TCP ada proses retransmisi maka pada saat paket dikirimkan, paket di copy untuk backup, seandanya paket yang dikirim ada masalah, maka digunakan untuk mengirmkan ulang. Dalam pemrograman socket ada 4 tahapan yang harus dilakukan :

- Open socket
- Menciptakan sebuah input data stream
- Menciptakan sebuah output data stream
- Close socket

**Open Socket, untuk sisi client**

```
Socket myclient = null;
try {
        myclient = new Socket("host",portNumber);
} catch (UnknownHostException e){
        e.printStackTrace();
} catch (IOException e){
        e.printStackTrace();
}
```

**Open socket untuk sisi Server**

```
ServerSocket myserver = null;
try {
        myserver = new ServerSocket(portNumber);
} catch (IOException e){
        e.printStackTrace();
}
```

Ketika kita mengimplementasikan sebuah server, kita butuh menciptakan objek dari ServerSocket dalam mode mendengar, guna menunggu koneksi dari client. Potongan programnya adalah

```
Socket servis= null;
try {
        servis = myserver.accept();
} catch (IOException e){
        e.printStackTrace();
}
```

**Menciptakan Input Stream**

Menciptakan sebuah input stream pada sisi client, kita dapat menggunakan BufferedReader class untuk menerima resposn dari sisi server. Potongan kodenya sebagai berikut :

```
BufferedReader is = null;
try {
        is = new BufferedReader(new InputStreamReader(myclient.getInputStream()));
} catch (IOException e){
        e.printStackTrace();
}
```

Sedangkan untuk input stream pada sisi server, kita juga dapat menggunakan BufferedReader untuk menerima input dari sisi client. Potongan kodenya

```
BufferedReader is = null;
try {
        is = new BufferedReader(new InputStreamReader(servis.getInputStream()));
} catch (IOException e){
        e.printStackTrace();
}
```

**Menciptakan output stream**

Pada sisi client, kita dapat menciptakan output stream untuk mengirimkan data ke server socket menggunakan DataOutputStream class. Berikut kode programnya

```
DataOutputStream os = null;
try {
                os = new DataOutputStream(myclient.getOutputStream());
}catch (IOException e){
        e.printStackTrace();
}
```

Sedangkan pada sisi server, kita juga dapat menggunakan class DataOutputStream untuk mengirimkan data ke client. Berikut kode programnya

```
DataOutputStream os = null;
try {
                os = new DataOutputStream(servis.getOutputStream());
}catch (IOException e){
        e.printStackTrace();
}
```

**Closing Socket**

Dalam setiap membuat suatu socket programming, tahapan pertama open dan tahapan terakhir adalah closing, kedua hal tersebut harus dilakukan. Berikut kode untuk proses penutupan/closing pada sisi client

```
try {
        os.close();
        is.close();
} catch (IOException e){
        e.printStackTrace();
}
```

Pada sisi servernya :

```
try {
        os.close();
        is.close();
        servis.close();
} catch (IOException e){
        e.printStackTrace();
}
```

## C. Praktikum

***Socket Server :***

```
package modul1;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.*;
public class simpleServer {
```

```
        public final static int port = 5000;
        public static void main(String[] args) {
                ServerSocket checkServer = null;
                String line;
                BufferedReader is  = null;
                DataOutputStream os = null;
                Socket clientSocket = null;
                try {
                        checkServer = new ServerSocket(port);
                        System.out.println("Simple server start.....");
                } catch(IOException e){
                        System.out.println(e);
                }
                //open input dan output stream
                try {
                        clientSocket = checkServer.accept();
                        is = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                        os = new DataOutputStream(clientSocket.getOutputStream());
                } catch(IOException e){
                        e.printStackTrace();
                }
                //Receive from client
                try {
                        line = is.readLine();
                        System.out.println(" We received :"+line);
                        if (line.compareTo("Greetings")==0){
                                os.writeBytes("..... and Solutions");
                        } else {
                                os.writeBytes("Sorry, you don't speak my protocol");
                        }
                }catch (IOException e){
                        e.printStackTrace();
                }
                //closing
                try {
                        os.close(); is.close();
                        clientSocket.close();
```

```
                } catch (IOException e){
                        e.printStackTrace();
                }
        }
}
```

***Socket Client***

```
package modul1;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.UnknownHostException;

public class simpleClient {
        public static final int port = 5000;
        public static void main(String[] args) {
                Socket cl =null;
                BufferedReader is = null;
                DataOutputStream os = null;
                BufferedReader st = new BufferedReader(new
InputStreamReader(System.in));
                String userinput = null;
                String output = null;
                try{
                        cl = new Socket("Machine remote",port);
                        is = new BufferedReader(new
InputStreamReader(cl.getInputStream()));
                        os = new DataOutputStream(cl.getOutputStream());
                } catch (UnknownHostException e){
                        System.out.println("Unknown Host : "+ e);
                }catch (IOException e0){
                        e0.printStackTrace();
                }
                //get input from user and send to the server
                try {
                        System.out.print("Please input a keyword : ");
                        userinput = st.readLine();
```

```
                os.writeBytes(userinput+"\n");
            } catch(IOException e){
                e.printStackTrace();
            }
            //receive from server
            try {
                output = is.readLine();
                System.out.println("Got from server : "+output);
            }catch(IOException e){
                e.printStackTrace();
            }
            //close
            try {
                is.close();
                os.close();
                cl.close();
            }catch(IOException e){
                e.printStackTrace();
            }
        }

}
```

Setelah kedua socket terbangun, maka langkah yang dilakukan adalah :

1. Kompilasi kedua socket tersebut
   Javac simpleServer.java
   Javac simpleClient.java
2. Kemudian jalankan (harus yang server terlebih dahulu)
   Java simpleServer
   Java simpleClient
3. Kemudian inputkan data pada aplikasi simpleClient yang sedang running
4. Amati keluaran pada sisi server dan sisi client.

## D. Latihan

1. Buatlah socket programming client server untuk proses penjumlahan, maksudnya adalah buat suatu socket server yang didalamnya terdapat suatu method untuk melakukan penjumlahan suatu data, pada socket client mengirimkan data yang akan di jumlahkan, dan socket server malakukan proses penjumlahan dan mengirimkan hasilnya ke socket client.

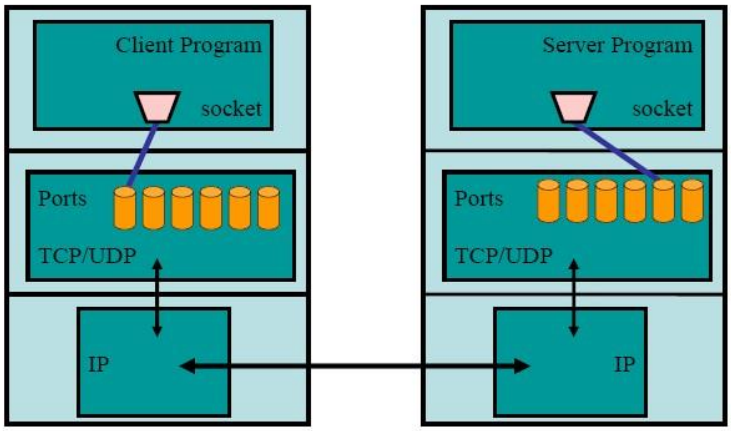
# MODUL 2
# PEMROGRAMAN SOCKET DENGAN PAKET UDP
## *(Menggunakan Pemrograman Java)*

## A. Tujuan

1. Mahasiswa mengenal kosep komunikasi client-server dengan Datagram socket
2. Mahasiswa mampu membuat aplikasi socket dengan paket UDP

## B. Pendahuluan

Dalam socket programming dapat dibangun dengan menggunakan dua model paket yaitu TCP (untuk *connection oriented*) dan UDP (untuk *connectionless*). Jika digambarkan komunikasi socket server dan socket client dengan jenis paketnya sebagai berikut




Gambar 2. Socket to socket communication

### UDP (*User Datagram Protocol*)

UDP merupakan protocol transport yang sederhana. Berbeda dengan TCP yang *connection oriented*, UDP bersifat *connectionless*. Dalam UDP tidak ada sequencing (pengurutan kembali) paket yang datang, *acknowledgement* terhadap paket yang datang, atau retransmisi jika paket mengalami masalah di tengah jalan.

Kemiripan TCP dengan UDP adalah pada penggunaan port number, sebagaimana digunakan pada TCP, UDP menggunakan port number ini membedakan pengiriman datagram ke beberapa aplikasi berbeda yang terletak pada komputer yang berbeda. Karena sifatnya yang *connectionless* dan *ureliable*, UDP digunakan oleh aplikasi aplikasi yang secara periodic melakukan aktivitas tertentu.

Permasalahan dalam komunikasi UDP, pertama ukuran pesan, proses penerimaan perlu menyediakan sejumlah elemen array untuk menerima sebuah pesan. Jika pesan terlalu besar akan dipotong sesuai dengan ukuran array. Kedua blocking, socket

menyediakan non-blocking send dan blocking receive untuk komunikasi datagram. Ketiga timeout, operasi blocking untuk receive cocok untuk menunggu permintaan client. Namun sangat tidak baik jika berada dalam keadaan blocking terus pada saat menunggu balasan, maka diperlukan timeout.

## UDP Socket (Datagram)

UDP socket diciptakan dengan menggunakan class DatagramSocket. Sebuah class DatagramSocket melakukan send dan receive data menggunakan packet, yang direpresentasikan dalam object DatagramPacket. Dalam suatu kasus dua buah program yang berkomunikasi dengan menggunakan koneksi UDP, keda program tersebut harus mempunyai koneksi DatagrapSocket pada suatu port dalam suatu komputer. Ini dapat diselesaikan dengan membuat objek DatagramSocket seperti berikut

```
DatagramSocket coonect = new DatagramSocket(5555);
```

Dalam contoh tersebut kita sedang membuat suatu koneksi UDP Socket dengan spesifik portnya 5555 pada local komputer. Jika kita tidak tahu port mana yang tidak sedang digunakan/tidak yakin bahwa port tersebut tidak sedang digunakan, maka dapat dilakukan dengan cara :

```
DatagramSocket ds = new DatagramSocket();
```

Dalm kasus ini kita tidak menggunakan port pada komputer local. Kita dapat mencari port yang sedang digunakan dengan cara

```
int portNumber = ds.getLocalPort();
```

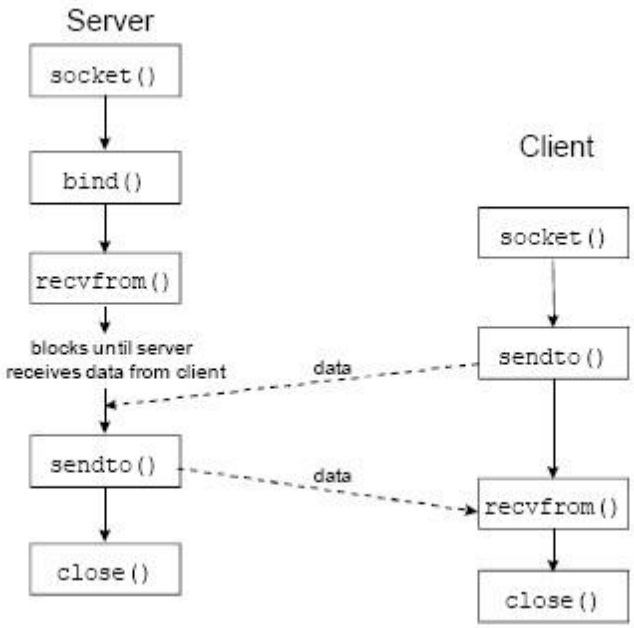
Data dalam komunikasi paket UDP dikirimkan diatas sebuah DatagramSocket dengan menggunakan DatagramPacket, dan setiap DatagramSocket mempunyai sebuah buffer data, alamat komputer remote yang akan dikirimi data dan nomer port yang digunakan server untuk listening. Untuk suatu client yang mengirimkan data ke server yang sedang listening pada port 5555 dan pada komputer/host "maswi", kita dapat menuliskan listing programnya sebagai berikut :

```
byte buf[] ={"h","e","l","l","o"};
InetAddress address = InetAddress.getByName("maswi");
DatagramPacket paket = new DatagramPacket(buf,buf.length,address,5555);
ds.send(paket);
```

Sedangkan untuk remote servernya adalah

*byte buf[] = new byte[256];*
*DatagramPacket paket = new DatagramPacket(buf,buf.length);*
*sd.receive(paket);*

Komunikasi client server untuk Datgram dapat digambarkan sebagai berikut :




## C. Praktikum

Ketik listing program tersebut dengan menggunakan IDE untuk java :

***Socket Server :***

```
package udp;
import java.net.*;
import java.io.*;
public class UDPServer{
public static void main(String args[]){
        DatagramSocket aSocket = null;
          try{
                aSocket = new DatagramSocket(6789);
                byte[] buffer = new byte[256];
                System.out.println("Server Running..... ");
                while(true){
                   DatagramPacket request = new DatagramPacket(buffer, buffer.length);
```

```
                aSocket.receive(request);
                DatagramPacket reply = new DatagramPacket(request.getData(),
                        request.getLength(), request.getAddress(), request.getPort());
                aSocket.send(reply);
            }
        }catch (SocketException e){
                System.out.println("Socket: " + e.getMessage());
        }catch (IOException e) {
                System.out.println("IO: " + e.getMessage());
        }finally {
                if(aSocket != null)
                aSocket.close();
        }
    }
}
```

***Socket Client***

```
package udp;
import java.net.*;
import java.util.Scanner;
import java.io.*;
public class UDPClient{
    public static void main(String args[]){
        // args give message contents and server hostname
        DatagramSocket aSocket = null;
        try {
                aSocket = new DatagramSocket(); Scanner sc = new
                Scanner(System.in); System.out.print("Inputkan message
                yang akan di submit : "); String message = sc.nextLine();
                byte [] m = message.getBytes();
                InetAddress aHost = InetAddress.getByName("127.0.0.1");
                int serverPort = 6789;
                DatagramPacket request = new DatagramPacket(m,  message.length(),
aHost, serverPort);
                aSocket.send(request);
                byte[] buffer = new byte[m.length];
                DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
                aSocket.receive(reply);
                System.out.println("Reply: " + new String(reply.getData()));
        }catch (SocketException e)
        {
                System.out.println("Socket: " + e.getMessage());
        }catch (IOException e)
        {
```

```
                System.out.println("IO: " + e.getMessage());
            }
        finally {
            if(aSocket != null)
                aSocket.close();
            }
    }
}
```

Setelah kedua socket terbangun, maka langkah yang dilakukan adalah :

1. Kompilasi kedua socket tersebut
   Javac UDPServer.java
   Javac UDPClient.java
2. Kemudian jalankan (harus yang server terlebih dahulu)
   Java UDPServer
   Java UDPClient
3. Kemudian inputkan data pada aplikasi UDPClient yang sedang running
4. Amati keluaran pada replay dari server pada aplikasi client.

## D. Latihan

1. Buatlah socket programming client server untuk proses penjumlahan, maksudnya adalah buat suatu socket server yang didalamnya terdapat suatu method untuk melakukan penjumlahan suatu data bertype array integer, pada socket client mengirimkan data yang akan di jumlahkan, dan socket server malakukan proses penjumlahan dan mengirimkan hasilnya ke socket client.

# MODUL 3
# PEMROGRAMAN SOCKET DENGAN KONSEP MULTI-THREADING
***(Menggunakan Pemrograman Java)***

## A. Tujuan

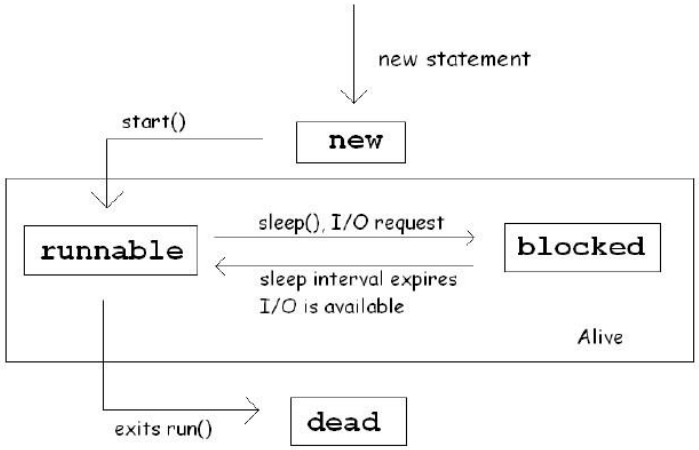
1. Menjelaskan mengenai kosep pemrograman Multi-threading
2. Membangun aplikasi socket dengan paket TCP/UDP dengan konsep multi-threading

## B. Pendahuluan

### Pemrograman Thread

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.

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 :



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.
2. *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.
3. *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.
4. *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.

**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 .............
        }
}
```

**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 ){ }
```

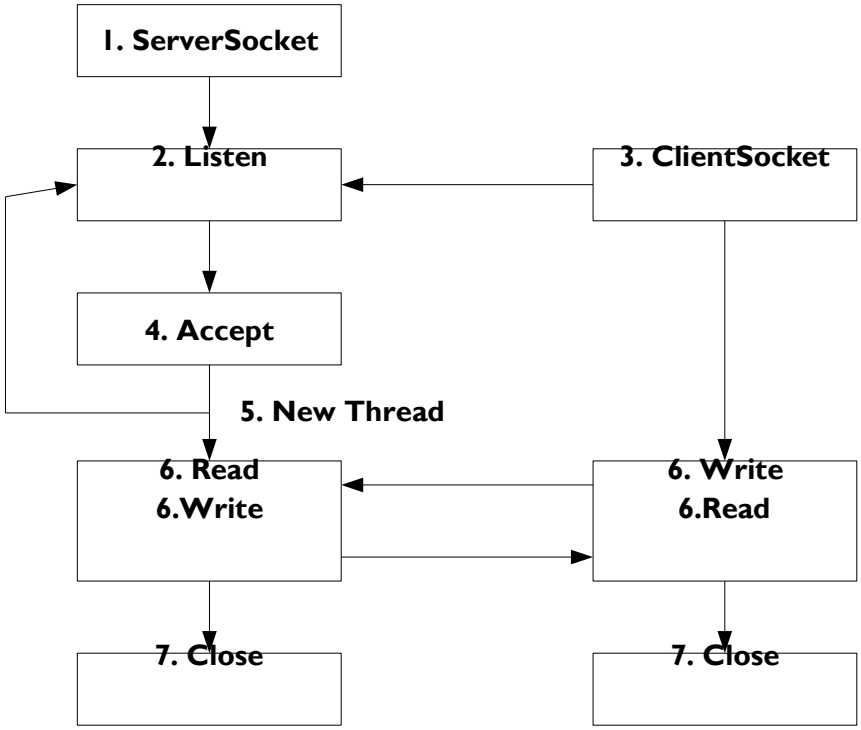
Dalam pembuatan aplikasi nantinya kita mungkin akan menggunakan beberapa thread untuk mengakses resource yang sama. Untuk menghindari masalah pada satu pengaksesan resource yang sama antar thread, maka satu thread harus tahu akan thread yang lain. Inilah yang disebut dengan synchronization. Struktur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

[public] **synchronized** [type_data] namaMethod (parameter masukkan){

```
        //koding.....
}
```

**Penggunaan Multithreading 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




## C. Praktikum

Praktikum 1 :

```
package th;
public class myThread implements Runnable {
        private static int i=0;
        @Override
        public void run() {
                while(i<100){
                        System.out.print(i+" , ");
                        i++;
                }
        }
}
```

Berikutnya contoh pemanggilan clas myThread tersebut :

```
package th;
public class myThreadTest {
	public static void main(String[] args) {
		myThread ot = new myThread();
		Thread th = new Thread(ot);
		th.start();
		for(int j=0;j<10;j++){
			System.out.println("From Main "+j);
		}
	}
}
```

Praktikum 2:

```
package th;
public class myThread2 implements Runnable {
	private static int i=0;
	Thread myth;
	public myThread2(){
		myth = new Thread(this);
		myth.start();
	}
	@Override
	public void run() {
		while(i<100){
			System.out.print(i+", ");
			i++;
		}
	}
}
```

Buat program/method main untuk menjalankan thread tersebut.

Praktikum 3 :

```
package th;
public class myThread3 extends Thread {
	private static int i=0;
	public void run(){
		while(i<100){
			System.out.print(i + " , ");
			i++;
		}
	}
}
```

Buatlah program untuk menjalankan thread teraebut.

Praktikum 4 :

```
package th;
public class myThreadTest2 {
	public static void main(String[] args) {
		myThread ot  = new myThread();
		Thread th = new Thread(ot);
		th.start();
		for(int j=0;j<10;j++){
			if (j==5){
				try {
					th.sleep(1000);
				} catch(Exception e){
					e.printStackTrace();
				}
			}
			System.out.println("From Main "+j);
		}
	}
}
```

Amati cara kerja dari program tersebut !.

Praktikum 5 :

```
package th;
public class mhThreadTest3 {
	public static void main(String[] args) {
		boolean isSuspend = false;
		myThread ot = new myThread();
		Thread th = new Thread(ot);
		th.start();
		for(int j=0;j<10000;j++){
			if (j%250 ==0){
				if (isSuspend){
					th.resume();
					System.out.println("Setelah resume i : "+ot.getI());
				} else {
					th.suspend();
					System.out.println("Setelah suspend i : "+ot.getI());
				}
				isSuspend = ! isSuspend;
			}
		}
		th.stop();
```

```
        }
}
```

Praktikum 6 : (Sebelum menggunakan synchronized)

```
package ok;
public class TwoStrings {
        static void print(String str1,  String str2) {
                System.out.print(str1);
                try {
                        Thread.sleep(500);
                } catch (InterruptedException ie) {}
                System.out.println(str2);
        }
}
```

Method print(String str1, String str2) akan diakses oleh lebih dari satu thread, sedangkan program yang terdiri lebih dari satu thread yang akan mengakses method tersebut adalah

```
package ok;
public class PrintStringsThread implements Runnable {
        Thread thread;
          String str1, str2;
          PrintStringsThread(String str1, String str2) {
            this.str1 = str1;
            this.str2 = str2;
            thread = new Thread(this);
            thread.start();
          }
          public void run() {
            TwoStrings.print(str1, str2);
          }
          public static void main(String args[]) {
                    new PrintStringsThread("Hello ", "there.");
                    new PrintStringsThread("How are ", "you?");
                    new PrintStringsThread("Thank you ",  "very much!");
                }

}
```

Kemudian Anda jalankan, dan amati hasilnya

Praktikum 7 : (menggunakan synchronized)

```
package ok;
public class TwoStrings {
        synchronized static void print(String str1,  String str2) {
                System.out.print(str1);
                try {
                        Thread.sleep(500);
                } catch (InterruptedException ie) {}
                System.out.println(str2);
        }
}
```

Kemudian program yang menggunakan method dalam synchronized adalah sama dengan praktikum 6, coba Anda amati outputnya, kemudian anda cerna konsep synchronization tersebut.

Praktikum 8 : (Penggunaan pemrograman Multi-threading pada socket programming dengan paket TCP)
Class turuna dari class Thread :

```
package tcp;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.IOException;
import java.net.Socket;
public class Connection extends Thread {
        DataInputStream in;
        DataOutputStream out;
        Socket clientSocket;
        public Connection (Socket aClientSocket) {
           try {
                clientSocket = aClientSocket;
                in = new DataInputStream( clientSocket.getInputStream());
                out =new DataOutputStream( clientSocket.getOutputStream());
                this.start();
            } catch(IOException e)  {System.out.println("Connection:"+e.getMessage());}
        }
        public void run(){
           try {                                        // an echo server
                String data = in.readUTF();
                out.writeUTF(data);
           } catch(EOFException e) {
                System.out.println("EOF:"+e.getMessage());
           } catch(IOException e) {
```

```
                System.out.println("IO:"+e.getMessage());
            } finally{
                    try {
                            clientSocket.close();
                    }catch (IOException e){}
                }
        }
}
```

Kemudian TCP Servernya :

```
package tcp;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
public class TCPServer {
        public static void main(String[] args) {
                try{
                        int serverPort = 7896;
                        ServerSocket listenSocket = new ServerSocket(serverPort);
                        System.out.println("Server sedang running.....");
                        while(true) {
                                Socket clientSocket = listenSocket.accept();
                                Connection c = new Connection(clientSocket);
                        }
                } catch(IOException e) {
                        System.out.println("Listen :"+e.getMessage());
                }
        }
}
```

Kemudian TCP Clientnya :

```
package tcp;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.Scanner;
public class TCPClient {
        public static void main(String[] args) {
                Socket s = null;
```

```
                try{
                    int serverPort = 7896;
                    String host="127.0.0.1";//host server, silahkan diganti jika server
tidak lokal
                    s = new Socket(host, serverPort);
                    DataInputStream in = new DataInputStream( s.getInputStream());
                    DataOutputStream out =       new DataOutputStream(
s.getOutputStream());
                    Scanner sc = new Scanner(System.in);
                    System.out.print("Inputkan Data yang akan di submit : ");
                    String input = sc.nextLine();
                    out.writeUTF(input);
                    String data = in.readUTF();
                    System.out.println("Received: "+ data) ;
                }catch (UnknownHostException e){
                    System.out.println("Sock:"+e.getMessage());
                }catch (EOFException e){
                    System.out.println("EOF:"+e.getMessage());
                }catch (IOException e){
                    System.out.println("IO:"+e.getMessage());
            }finally {
                    if(s!=null)
                            try {
                                    s.close();
                            }catch (IOException e){
                                    System.out.println("close:"+e.getMessage());
                                    }
                    }
        }
}
```

Kemudian Anda compile TCPClient dan TCPServer, setelah itu running untuk TCPServer baru kemudian Anda running TCPClientnya. Amati hasil dan prosesnya.

## D. Latihan

1. 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.
2. Buatlah untuk kasus yang sama dengan paket Datagram (UDP).

# MODUL 4
# HTTP (HyperText Transport Protocol)
## *(Menggunakan Pemrograman Java)*

## A. Tujuan

1. Memberikan gambaran tentang konsep pemrograman request-response pada HTTP
2. Mampu membangun server HTTP

## B. Pendahuluan

### HTTP

HTTP merupakan hypertext transport protocol (RFC 1945) adalah protocol yang digunakan untuk komunikasi antara web browser dengan web sserver. HTTP merupakan implementasi dari protocol TCP, dan menggunakan port 80 sebagai defaultnya. HTTP bersifat stateless, artinya tidak ada informasi yang disimpan, selain itu juga bersifat request-response. HTTP client (user agent misalnya) mengirimkan permintaan (request) ke HTTP server dan server meresponse sesuai dengan permintaan/request HTTP client. User agent software yang bertindak sebagai HTTP client (yang melakukan request) contohnya seperti browser Mozila, Netscape, Microsoft Internet Explorer atau software browser yang berbasis text seperti Lynx.

HTTP pertama kali di luncurkan mengguanakan versi 1.0, sekarang disempurnakan menjadi very 1.1. Perbedaan HTTP/1.0 dengan HTTP/1.1 :

- HTTP/1.0 membuka satu koneksi untuk setiap permintaan URI
  Header = Connection : close
- HTTP/1.1 dapat menggunakan sebuah koneksi TCP untuk beberapa permintaan URI(persistent)
  Header = Connection: Keep-Alive
  Kecuali jika client menyatakan tidak menggunakan hubungan persisten (header=Connection: close)

HTTP status codes

- 200 OK
- 201 created
- 202 accepted
- 204 no content
- 301 moved perm
- 302 moved temp
- 304 not modified
- 400 bad request

- 401 unauthorized
- 403 forbidden
- 404 not found
- 500 int. server error
- 501 not impl.
- 502 bad gateway
- 503 svc not avail

## C. Praktikum

Dalam praktikum kali ini kita akan coba bangun sebuah web server yang sederhana untuk melayani request dan response suatu client. Client nantinya hanya menggunakan software browser. Langka-langkah yang harus dilakukan adalah :

1. Membuat class HTTPResponse, class ini malakukan proses untuk meresponse request client.

```
package server;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.Date;
public class HTTPResponse {
	private byte[] content=null;
	private String contentType="text/html";
	private DataOutputStream client = null;
	private Socket clientSock=null;
	private int httpCode = 200;
	private int contentLength=0;
	private String CRLF ="\r\n";
	public HTTPResponse(int responseCode, Socket sock){
		this.clientSock = sock;
		this.httpCode = responseCode;
	}
	public void setContentType(String type){
		this.contentType = type;
	}
	public String getContentType(){
		return this.contentType;
	}
	private String buildResponsHeader(byte[] content){
		int content_length= content.length;
		String content_type = this.getContentType();
		Date today = new Date();
```

```
                String expire_date = today.toString();
                String header = "HTTP "+httpCode+" OK\r\nServer:Personal Web
server\r\n";
                header = header +"Content-Type: "+content_type+"\r\nContent-Length:
"+content_length+"\r\n";
                header = header + "Connection: close\r\nExpires:
"+expire_date+"\r\n\r\n";
                return header;
        }
        public void write(byte[] writeme) throws IOException {
                client = new DataOutputStream(clientSock.getOutputStream());
                client.write(buildResponsHeader(writeme).getBytes());
                client.write(writeme);
                client.flush();
                client.close();
                clientSock.close();
        }
}
```

Setelah anda tulis coding tersebut kemudian compile, kemudian ikuti langkah berikutnya.

2. Membuat class HTTPRequest, class ini berfungsi untuk melakukan proses terhadap request yang dilakukan oleh client.

```
package server;
import java.io.ByteArrayOutputStream;
import java.util.Enumeration;
import java.util.Hashtable;
import java.util.StringTokenizer;
public class HTTPRequest {
        private Hashtable headers = new Hashtable();
        private Hashtable parameters = new Hashtable();
        private ByteArrayOutputStream out = new ByteArrayOutputStream();
        private byte[] content = null;
        private String requestURI="";
        private String method ="";
        public HTTPRequest(byte[] headers){
                this.parseHeaders(headers);
        }
        private void parseHeaders(byte[] rawHeaders){
                int eoh = findEOH(rawHeaders);
                String heads = new String(rawHeaders,0,eoh);
                StringTokenizer st = new StringTokenizer(heads,"\r\n");
                StringTokenizer st_firstLine = new StringTokenizer(st.nextToken()," ");
                this.method = st_firstLine.nextToken().trim();
                this.requestURI = st_firstLine.nextToken().trim();
```

```
                while (st.hasMoreTokens()){
                        String requestLine = st.nextToken();
                        int separator = requestLine.indexOf(": ");
                        String header = requestLine.substring(0,separator);
                        String value =
requestLine.substring(separator+1,requestLine.length());
                        headers.put(header.trim(), value.trim());
                }
                if ( this.getMethod().equals("POST")){
                        this.content = new
byte[Integer.parseInt(this.getHeader("Content-Length"))];
                        System.arraycopy(rawHeaders,eoh+4, this.content, 0,
this.content.length);
                } else{
                        System.out.println(new String(rawHeaders));
                }

        }
        public static int findEOH(byte[] headers){
                String heads = new String(headers);
                return heads.indexOf("\r\n\r\n");
        }
        public String getRequestURI(){
                return this.requestURI;
        }
        public String getMethod(){
                return this.method;
        }
        public String getParamater(String name){
                return (String) this.parameters.get(name);
        }
        public String getHeader(String name){
                return (String) this.headers.get(name);
        }
        public Enumeration getHeaders(){
                return headers.elements();
        }
        public byte[] getContent(){
                return this.content;
        }
}
```

Ketik, kemudian compile, stelah itu kemudian kerjakan tahapan berikutnya :

3. Membuat class HTTPClient, class ini merupakan class yang bertugas melakukan proses request dan response, jadi dalam class ini memenafaatkan dua class yang telah dibuat yaitu HTTPRequest dan HTTPResponse. Berikut classnya

```
package server;
import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.net.Socket;
public class HTTPClient extends Thread {
	Socket socket;
	ByteArrayOutputStream bos = new ByteArrayOutputStream();
	DataOutputStream out = null;
	public HTTPClient(Socket socket){
		this.socket = socket;
		setPriority(NORM_PRIORITY-1);
		start();
	}
	public void run(){
		try {
			BufferedReader requestReader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
			ByteArrayOutputStream requestBA = new
ByteArrayOutputStream();
			int rch = socket.getInputStream().available();
			System.out.println(rch);
			while(rch-->0){
				requestBA.write(requestReader.read());
			}
			HTTPRequest request = new
HTTPRequest(requestBA.toByteArray());
			File resource = new
File("C:/wamp/www/kuliah"+request.getRequestURI());
			FileInputStream fis = new FileInputStream(resource.getPath());
			byte[] data = new byte[(int) resource.length()];
			fis.read(data,0,data.length);
			System.out.println(new String(data).toString());
			HTTPResponse response = new HTTPResponse(200,socket);
			response.setContentType("text/html");
			response.write(data);
		} catch(Exception e){
```

Silahkan disesuakan dengan path yang diinginkan

```
                    e.printStackTrace();
                }
        }
}
```

Ketik dan compile, kemudian kita lanjut ke langkah terakhir ini

4. Membuat class HTTPServer, class ini berisi socket yang bertugas

```
package server;
import java.io.IOException;
import java.net.ServerSocket;
public class HTTPServer implements Runnable {
        private ServerSocket serversocket = null;
        public static void main(String[] args){
                new HTTPServer();
        }
        public HTTPServer(){
                Thread server_thread = new Thread(this);
                server_thread.start();
        }
        public void run(){
                try {
                        System.out.print("Server is Running.....");
                        serversocket = new ServerSocket(8080);
                } catch(IOException e){
                        e.printStackTrace();
                }
                while (true){
                        try {
                                new HTTPClient(serversocket.accept());
                        } catch (Exception e){
                                e.printStackTrace();
                        }
                }
        }
}
```

Ketik dan compile. Setalh itu kemudian jalankan HTTPServer tersebut dengan perintah : Java HTTPServer

Setalah server dalam keadaan running maka kita dapat membuka browser kemudian ketik URL : http://localhost:8080/index.html. Maka pada browser akan menampilkan file index.html tersebut layaknya kita mengakses web server apache.

## D. Latihan

1. Coba Anda modifikasi program tersebut sehingga untuk melakukan request adalah cukup dengan perintah http://localhost:8080/ .
2. Coba anda gabungkan anda modifikasi sehingga server HTTP tersebut dapat mengeksekusi file PHP (atau intinya kita bisa membuat suatu aplikasi web dengan form input dan dapat memproses kemudian menampilkan kembali ke browser).

# MODUL 5
# OBJECT OVER SOCKET (*Distributed Object*)
## *(Menggunakan Pemrograman Java)*

## A. Tujuan

1. Memberikan gambaran tentang konsep persistent object pada socket
2. Mampu membangun aplikasi object terdistribusi dengan menggunakan socket programming

## B. Pendahuluan

### Object Serialisasi

Object serialisasi adalah mekanisme yang biasa digunakan dalam Remote Method Invocation (RMI), akan tetapi sebagaimana type data primitive memungkinkan object dapat dikirimkan melalui input/output stream, missal disimpan dalam file ataupun socket. Bahkan sekarang sudah berkembang database object seperti odb4o. Dalam bekerja dengan object serialisasi, suatu object java harus mengimplementasikan interface java.io.Serializable. Pada saat berbicara tentang object pasti kita juga akan berbicara bagaimana persistent data berupa object dengan menggunakan input/output stream. Dimana strukturnya/sintaknya :

```
FileOutputStream fos = new FileOutputStream("str.out");
ObjectOutputStream oos = new ObjectOutputStream(fos);
Oos.writeObject("Objek yang akan disimpan");
```

Method writeObject() adalah method yang digunakan untuk menyimpan object ke output stream. Method ini dapat dipanggil berulangkali pada saat mau digunakan untuk menyimpan object. Object yang akan disimpan harus mengimplementasikan interface java.io.Serializable.

Untuk proses pembacaan object yang telah disimpan, menggunakan method readObject(), method ini dapat dipanggil berulangkali untuk membaca sejumlah object yang telah disimpan dalam input stream.

```
FileInputStream fis = new FileInputStream("str.out");
ObjectInputStream ois = new ObjectInputStream(fis);
Object o = ois .readObject();
```

## C. Praktikum

Praktikum 1 : Buat class yang nantinya sebagai object yang akan disimpan

```
package data;
import java.io.Serializable;
public class Employee implements Serializable {
        private String name;
        private int age;
        private float salary;
        public Employee(String name, int age, float salary){
                this.name = name;
                this.age = age;
                this.salary = salary;
        }
        public void print(){
                System.out.println("Record untuk : "+this.name);
                System.out.println("Name : "+this.name);
                System.out.println("Age : "+this.age);
                System.out.println("Salary : "+ this.salary);
        }
}
```

Kemudian buat class yang digunakan untuk proses saved data berupa objek

```
package data;
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;
public class saveData {
public static void main(String[] args) {
        try {
                Employee sifa = new Employee("Sifa",28,2500000);
                Employee maswi = new Employee("Maswi",34,2800000);
                FileOutputStream fos = new FileOutputStream("db");
                ObjectOutputStream oos = new ObjectOutputStream(fos);
                oos.writeObject(sifa);
                oos.writeObject(maswi);
                oos.flush();
        } catch (Exception e){
                e.printStackTrace();
        }
  }
}
```

Kemudian silahkan di compile, kemudian jalankan. Langkah selanjutnya adalah membuat class yang bertugas untuk membaca data object yang telah disimpan dalam file.

```
package data;
import java.io.FileInputStream;
import java.io.ObjectInputStream;
public class readData {
	public static void main(String[] args) {
		try {
			FileInputStream fis = new FileInputStream("db");
			ObjectInputStream ois = new ObjectInputStream(fis);
			Employee dat1 = (Employee) ois.readObject();
			Employee dat2 = (Employee) ois.readObject();
			dat1.print();
			dat2.print();
		} catch(Exception e){
			e.printStackTrace();
		}
	}

}
```

Lakukan langkah yang sama yaitu compile dan jalankan. Hasilnya kurang lebih seperti ini :

*Record untuk : Sifa*
*Name : Sifa*
*Age : 28*
*Salary : 2500000.0*
*Record untuk : Maswi*
*Name : Maswi*
*Age : 34*
*Salary : 2800000.0*

Praktikum 2 : kita akan membuat object yang ditulis dalam socket (objek terdistribusi)

Buat class yang akan dipertukarkan, yaitu class Data yang nantinya akan dikirimkan dari client ke server, dan class matObjek yang nantinya akan dirimkan ari server ke client sebagai balasan.

```
package oversock;
import java.io.Serializable;
public class Data implements Serializable {
	private float a[]=null;
	private float b[]=null;
	public Data(float[] a, float[] b) {
		super();
		this.a = a;
		this.b = b;
	}
```

```
        public float[] getA() {
                return a;
        }
        public void setA(float[] a) {
                this.a = a;
        }
        public float[] getB() {
                return b;
        }
        public void setB(float[] b) {
                this.b = b;
        }
}
```

Kemudian class matObjek :

```
package oversock;
import java.io.Serializable;
public class matObjek implements Serializable  {
        private float hasil[]=null;
        public void jumlah(Data a){
                if (a.getA().length != a.getB().length){
                        return;
                } else {
                        float[] x = a.getA();
                        float[] y = a.getB();
                        hasil = new float[x.length];
                        for(int i=0;i<x.length;i++){
                                hasil[i] = x[i]+ y[i];
                        }
                }
        }
        public float[] getHasil(){
                return this.hasil;
        }
}
```

Langkah selanjutnya membuat aplikasi server :

```
package oversock;
import java.io.InputStream; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream; import
java.io.OutputStream; import
java.net.ServerSocket; import
java.net.Socket;
```

```
public class matServer {
        public static void main(String[] args) {
                ServerSocket server = null;
                Socket soket = null;
                Data x = null;
                matObjek y=null;
                float hasil[] = null;
                try {
                        System.out.print("Server is Running.....");
                        server = new ServerSocket(4343);
                        soket = server.accept();
                        InputStream is = soket.getInputStream();
                        ObjectInputStream ois = new ObjectInputStream(is);

                        OutputStream os = soket.getOutputStream();
                        ObjectOutputStream oos = new ObjectOutputStream(os);
                        x = (Data) ois.readObject();
                        y = new matObjek();
                        y.jumlah(x);
                        oos.writeObject(y);
                        oos.flush();
                         ois.close();
                         oos.close();
                } catch(Exception e){
                        e.printStackTrace();
                }
        }

}
```

Langkah yang terakhir adalah membuat aplikasi client :

```
package oversock;
import java.io.InputStream; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream; import
java.io.OutputStream; import
java.net.Socket;
public class matClient {
        public static void main(String[] args) {
                Data data = null;
                matObjek  asil = null;
                try {
                        Socket soket = new Socket("127.0.0.1",4343);
                        OutputStream os = soket.getOutputStream();
```

```
                    ObjectOutputStream oos = new ObjectOutputStream(os);
                    InputStream is = soket.getInputStream();
                    ObjectInputStream ois = new ObjectInputStream(is);
                    float[] a ={3,4,5,6,7};
                    float[] b ={5,6,7,8,9};
                    data = new Data(a,b);
                    oos.writeObject(data);
                    oos.flush();
                    asil = (matObjek) ois.readObject();
                    System.out.println("Hasil Penjumlahan : ");
                    for(int i=0;i<a.length;i++){
                            System.out.print(asil.getHasil()[i] + ", ");
                    }
                    oos.close();
                    ois.close();
                }catch(Exception e){
                    e.printStackTrace();
                }
        }
}
```

Kemudian lakukan compilasi, dan running, dengan urutan server terlebih dahulu baru kemudian clientnnya.

## D. Latihan

1. Buatlah sebuah database dalam database server MySQL 5.0 atau diatasnya, kemudian buat sebuah table, kemudian isi datanya. Buat socket server yang didalamnya terdapat proses koneksi dan query ke database berdasarkan permintaan dari client socket. Data yang dikirimkan dari client atau dari server berupa objek.

# MODUL 6
# KONEKSI DATABASE MYSQL 5.0 VIA SOCKET
## *(Menggunakan Pemrograman Java)*

## A. Tujuan

1. Memberikan gambaran tentang konsep koneksi ke database MySQL
2. Mampu membangun aplikasi object terdistribusi dengan menggunakan socket programming yang terkonkesi dengan database.

## B. Pendahuluan

Dalama membangun aplikasi yang memanfaatkan database dalam java ada beberapa hal yang perlu dilakukan instlasi yaitu :

1. Memasang Java dan JDBC.
2. Memasang sebuah driver pada mesin Anda
3. Memasang jika DBMS jika perlu

Setelah tahapan instalasi tersebut dilakukan pengaturan database, buat sebuah database di MySQL dengan nama missal : test. Tahapan selanjutnya adalah membangun koneksi. Dalam membangun koneksi perlu dilakukan dua langkah :

- Memuat driver, dalam memuat driver yang ingin anda gunakan caranya sangat sederhana dan hanya perlu satu baris code missal untuk database MySQL :
  Class.*forName*("com.mysql.jdbc.Driver");
- Membuat koneksi, dalam membuat koneksi kodingnya adalah
  String data = "jdbc:mysql://"+server+":3306/"+db;
  Connection con = DriverManager.*getConnection*(data,user,pswd);

Langkah berikutnya membuat table, untuk membuat table dapat digunakan tools yang lain suapaya tidak selalu bermain dalam koding, missal menggunakan Navicat, phpmyadmin. Setelah kita membuat table, kita dapat melakukan beberapa operasi seperti select, insert update atau delete. Untuk dapat melakukan operasi tersebut maka diperlukan beberapa koding sebelumnya yaitu :

Membuat Statement
Gunakan object Connection, yang dalam contoh
Connection con = DriverManager.*getConnection*(data,user,pswd);
Statement stm = con.createStatement();
Untuk menjalankan Statement dapat menggunakan 3 perintah berikut, tergantung dari apa yang mau dieksekusi

- Untuk perintah SELECT, maka menggunakan method executeQuery(String sql).
- Untuk perintah INSERT, UPDATE, DELETE, CREATE, DROP menggunakan method executeUpdate(String sql).

- Untuk stored procedure dengan hasil multiple dengan menggunakan method execute(String sql).

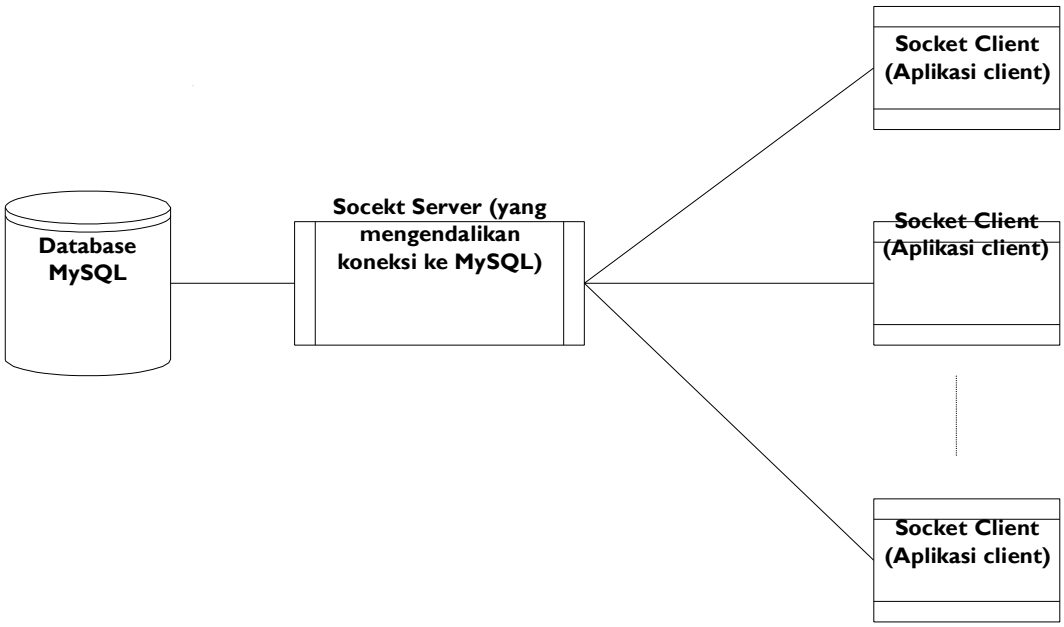
ResultSet
Sebuah objek ResultSet adalah objek java yang berisi hasil-hasil sewaktu menjalankan sebuah query SQL, hasilnya berupa baris-baris yang memenuhi kondisi query tersebut. Data yang disimpan dalam ResulsSet diambil melalui serangkain method get yang mengijinkan akses ke berbagai kolom dari baris. Contohnya :

```
ResultSet rs = stmt.executeQuery("SELECT * FROM mahasiswa");
while(rs.next()){
        System.out.println("Nim : " + rs.getInt(1));
        System.out.println("Nama : "+ rs.getString(2));
}
```

Perintah untuk mengambil data dari table mahasiswa pada kolom pertama menggunakan method getInt(1), Int menunjukkan type data dari data yang disimpan dalam table kolom pertama, sedangkan 1 menunjukkan kolom pertama. Kemudian getString(2), berarti untuk mengambil data pada kolom kedua, String menunjukkan type data yang tersimpan dalam kolom kedua.

## C. Praktikum

Praktikum 1 : Kita akan membangu aplikasi tree-tier berbasis socket programming, dengan model aplikasinya sebagai berikut :



Membuat class mahasiswa

```
package test;
import java.io.Serializable;
public class mahasiswa implements Serializable {
        private int nim;
        public int getNim() {
                return nim;
        }
        public void setNim(int nim) {
                this.nim = nim;
        }
        public String getNama() {
                return nama;
        }
        public void setNama(String nama) {
                this.nama = nama;
        }
        private String nama;

}
```

Kemudian membuat class kuliah

```
package test;
import java.io.Serializable;
import java.util.ArrayList;
public class kuliah implements Serializable {
        private ArrayList<mahasiswa> isi = null;
        public ArrayList<mahasiswa> getIsi() {
                return isi;
        }
        public void setIsi(ArrayList<mahasiswa> isi) {
                this.isi = isi;
        }
}
```

Langkah selanjutnya kita buat class yang bertuga untuk melakukan koneksi ke database MySQL 5.0, database yang akan dikoneksi kita beri nama **test** dan didalamnya ada table mahasiswa. Sedangkan user =root dan paswordnya = “”.

```
package test;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
import java.util.ArrayList;
public class conDatabase {
	private Connection con=null;
	private Statement stm = null;
	public conDatabase(String server, String db, String user, String pswd) {
		super();
		try {
			Class.forName("com.mysql.jdbc.Driver");
			String data = "jdbc:mysql://"+server+":3306/"+db;
			con = DriverManager.getConnection(data,user,pswd);
			stm = con.createStatement();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
	public ResultSet query(int nim) throws SQLException{
		String sql = "SELECT * FROM mahasiswa WHERE nim="+nim;
		return this.stm.executeQuery(sql);
	}
	public ArrayList<mahasiswa> getMahasiswa(int nim){
		ArrayList<mahasiswa> isi = new ArrayList<mahasiswa>();
		try {
			ResultSet rs = this.query(nim);
			while(rs.next()){
				mahasiswa aa = new mahasiswa();
				aa.setNim(rs.getInt(1));
				aa.setNama(rs.getString(2));
				isi.add(aa);
			}
		} catch (Exception e){
			e.printStackTrace();
		}
		return isi;
	}
	public void inputData(int a, String b) throws SQLException{
		String sql ="INSERT INTO mahasiswa values("+a+",'"+b+"')";
		stm.executeUpdate(sql);
	}
}
```

Langkah selanjutnya membuat class yang merupakan turunan dari class Thread untuk memanfaatkan class conDatabase, yang nantinya juga bertugas untuk malakukan proses atas permintaan dari client socket. Socket ini berbasis multithreading sehinggga dapat melayani banyak client.

```
package test;
import java.io.InputStream; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream; import
java.io.OutputStream; import
java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
public class connect extends Thread {
        Socket client;
        mahasiswa x = null;
        Connection con=null;
        Statement stm = null;
        siswa z= null;
        public connect(Socket client) {
                super();
                this.client = client;
                try {
                        InputStream is = client.getInputStream();
                        ObjectInputStream ois = new ObjectInputStream(is);
                        OutputStream os = client.getOutputStream();
                        ObjectOutputStream oos = new ObjectOutputStream(os);
                        x = (mahasiswa) ois.readObject();
                        conDatabase  test =
                        new conDatabase("localhost","test","root","");
                        test.inputData(x.getNim(), x.getNama());
                        ArrayList<mahasiswa> isi = test.getMahasiswa(x.getNim());
                        z = new kuliah();
                        z.setIsi(isi);
                        oos.writeObject(z);
                        oos.flush();
                        ois.close();
                        oos.close();
                } catch(Exception e){
                        e.printStackTrace();
                }
        }
}
```

Berikutnya membuat server socket :

```
package test;
import java.io.InputStream; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream; import
java.io.OutputStream; import
java.net.ServerSocket; import
java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
public class matServer extends Thread {
        public ServerSocket server = null;
        public matServer(){
                try {
                        System.out.print("Server is Running.....");
                        server = new ServerSocket(4343);
                        this.start();
                } catch(Exception e){
                        e.printStackTrace();
                }
        }
        public static void main(String[] args) {
                new matServer();
        }
        public void run(){
                try {
                        while(true){
                                Socket client = server.accept();
                                connect cc = new connect(client);
                        }
                } catch(Exception e){
                        e.printStackTrace();
                }
        }
}
```

Langkah yang terakhir membuat socket client :

```
package test;
import java.io.InputStream; import
java.io.ObjectInputStream; import
java.io.ObjectOutputStream; import
java.io.OutputStream;
```

```
import java.net.Socket;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;
public class matClient {
	public static void main(String[] args) {
		mahasiswa data = null;
		siswa  asil = null;
		try {
			Socket soket = new Socket("127.0.0.1",4343);
			OutputStream os = soket.getOutputStream();
			ObjectOutputStream oos = new ObjectOutputStream(os);
			InputStream is = soket.getInputStream();
			ObjectInputStream ois = new ObjectInputStream(is);
			data = new mahasiswa();
			Scanner sc  = new Scanner(System.in);
			System.out.print("Inputkan nim : ");
			data.setNim(sc.nextInt());
			System.out.print("Inputkan nama : ");
			data.setNama(sc.next());
			oos.writeObject(data);
			oos.flush();
			asil = (kuliah) ois.readObject();
			ArrayList<mahasiswa> oke = asil.getIsi();
			Iterator it = oke.iterator();
			while(it.hasNext()){
				mahasiswa ace = (mahasiswa) it.next();
				System.out.println("Nim : "+ ace.getNim());
				System.out.println("Nama : "+ ace.getNama());
			}

			oos.close();
			ois.close();
		}catch(Exception e){
			e.printStackTrace();
		}
	}

}
```

Kemudian Anda compile dan coba running, dengan running dulu untuk client socket baru kemudian server socket.

## D. Latihan

1. Modifikasi pada praktikum 1 sehingga dapat digunakan untuk proses DELETE dan UPDATE data pada table mahasiswa.

# Mengenal REST API - JSON

**Pendahuluan**


**REST** (*Representional State Transfer*) adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data dan metode ini sering diterapkan dalam pengembangan aplikasi, sedangkan **API** singkatan dari *Application Programming Interface* yaitu sekumpulan perintah, fungsi, serta protokol yang dapat digunakan oleh programmer saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan programmer untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi.

**JSON** (dibaca:jason) merupakan singkatan dari JavaScript Object Notation. JSON merupakan salah satu open standard untuk pertukaran data dalam bentuk teks. JSON mempunyai format sederhana, sehingga banyak digunakan untuk proses pertukaran data.

Berikut beberapa kelebihan dari JSON antara lain:

1. mudah dibaca atau ditulis
2. banyak bahasa pemrograman menyediakan library atau function yang memudahkan untuk membaca atau membuat struktur JSON
3. mudah dipetakan pada struktur data yang digunakan oleh sebagian besar bahasa pemrograman terkait data berupa number, string, boolean, null, array dan associative array.

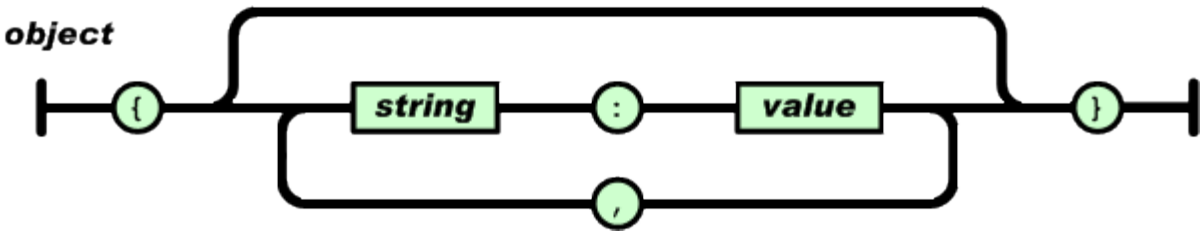
**Struktur JSON**

JSON dibangun di atas dua struktur, yaitu:

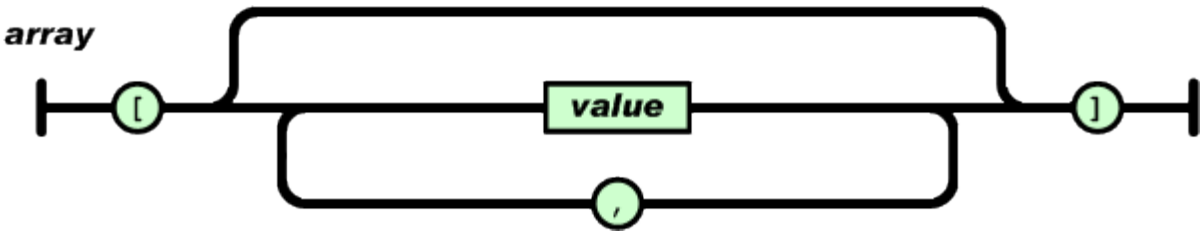
1. Kumpulan pasangan nama/nilai. Dalam bahasa pemerograman, hal ini dinyatakan sebagai objek (object), rekaman (record), struktur (struct), kamus (dictionary), tabel hash (hash table), daftar berkunci (keyed list), atau associative array.
2. Daftar nilai terurutkan (an orderes list of values). Dalam bahasa pemrograman, hal ini dinyatakan sebagai larik (array), vektor (vector), daftar (list) atau urutan (sequence).

Bentuk-bentuk JSON

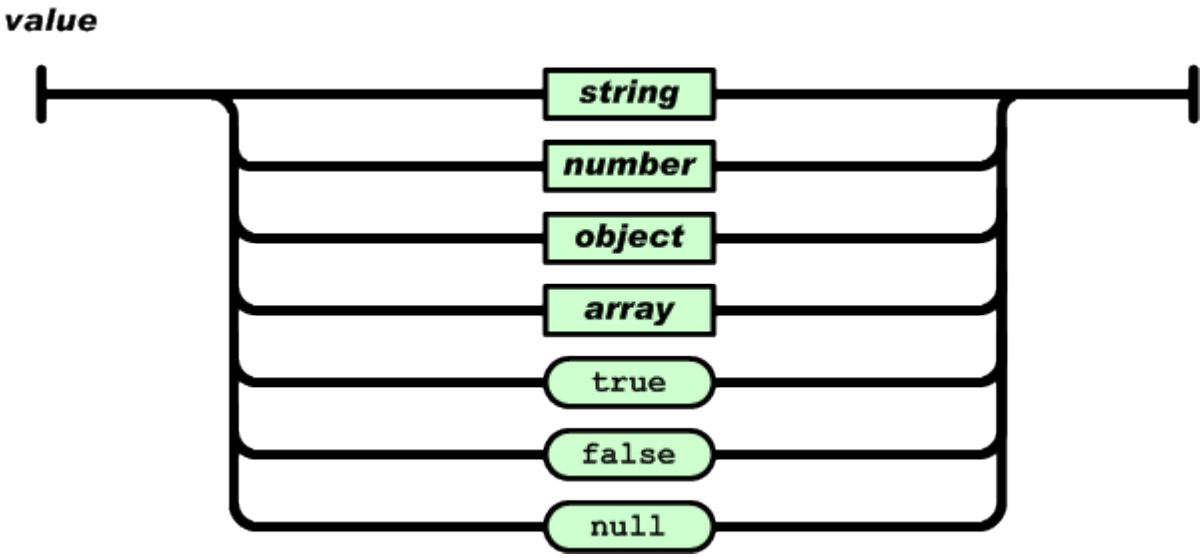
**Objek** adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma).



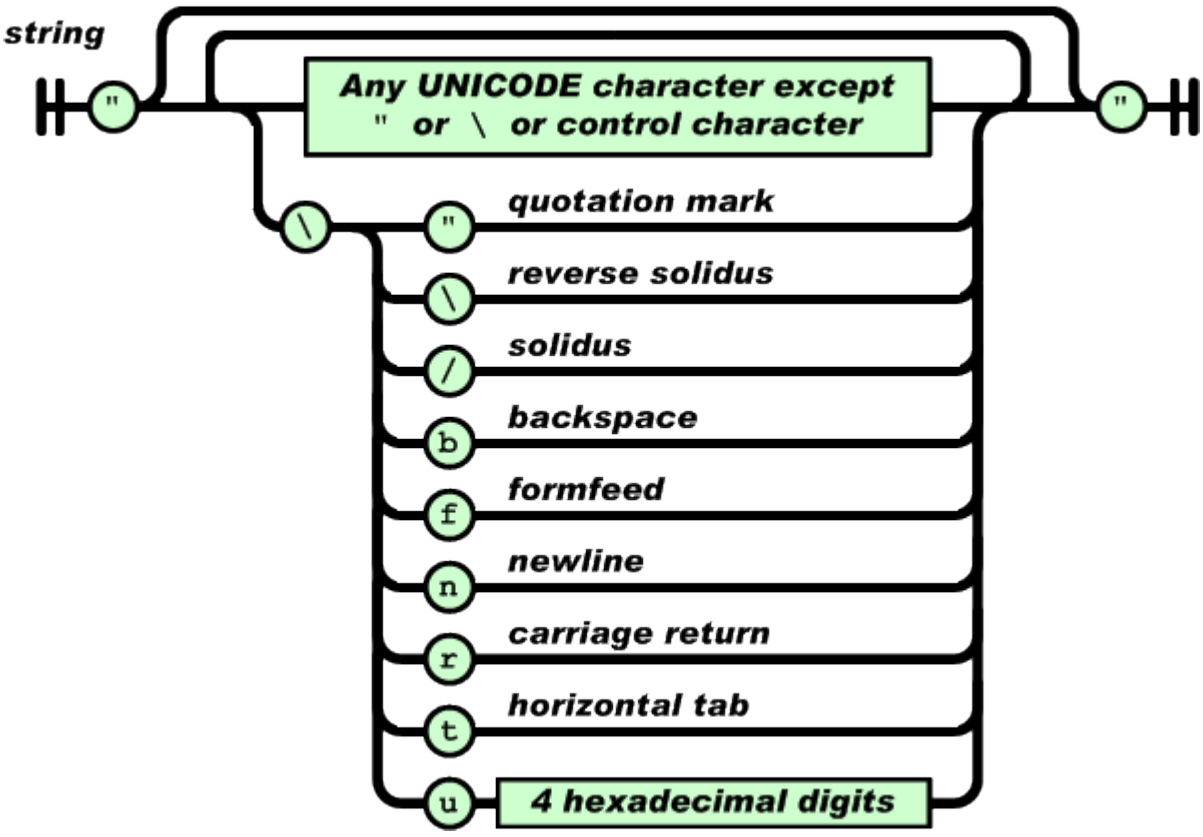

**Larik** adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [ (kurung kotak buka) dan diakhiri dengan ] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).



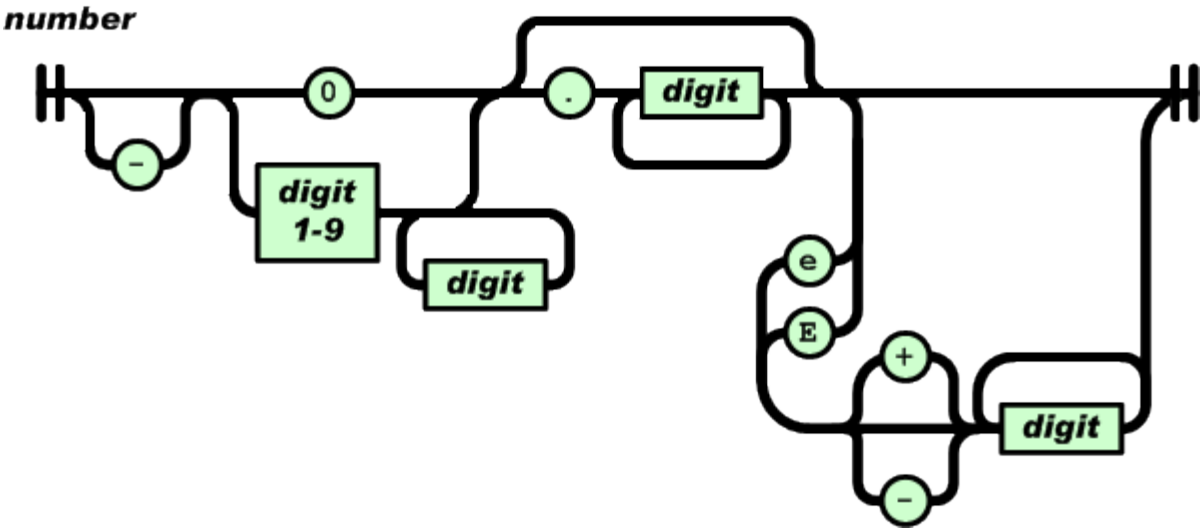

**Nilai** (*value*) dapat berupa sebuah **string** dalam tanda kutip ganda, atau *angka*, atau true atau false atau null, atau sebuah *objek* atau sebuah *larik*. Struktur-struktur tersebut dapat disusun bertingkat.




**String** adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.




**Angka** adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.




Spasi kosong (*whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan

**Contoh JSON**

{“mahasiswa” : {

“nama”: “Agus”,

“nim” : “M0509001”

},

“krs” : [

{“tahun”: 2017,”kodeMk”:”090101”,”tanggal_approve”:2017-01-01},

{“tahun”: 2017,”kodeMk”:”090102”,”tanggal_approve”:2017-01-01}

]

}

## Bahan Praktikum

Komunikasi Data Menggunakan JSON

1. **Data Array Sederhana**

**File jsonMahasiswaServer.php**

```
<?php

$DataMahasiswa=array();
$DataMahasiswa=array('nim'=>'M0516001','nama'=>' Iwan
Subagya','alamat'=>'Surakarta','tempatLahir'=>'Surakarta','tglLahir'=>'1997-01-01');
echo json_encode($DataMahasiswa);

?>
```

Disimpan di htdocs/sister.1/jsonMahasiswaServer.php
File ini merupakan file yang digunakan sebagai sumber data. Selanjutnya akan dibuat file untuk memanggil sebagai aplikasi client.

File **jsonMahasiswaClient.php**

```
<?php
$Data=file_get_contents('http://localhost/sister.1/jsonMahasiswaServer.php');
$Hasil=json_decode($Data);
echo 'Nama:'.$Hasil->nama;
echo '<br/>';
echo 'Tanggal Lahir: '.$Hasil->tglLahir;

?>
```

Disimpan di folder htdocs/sister.1

2. **Data Array Multidimensi**

File **jsonMultidimensiServer.php**

```
<?php
```

```
        $KRS=array();
        $MK=array(
                        array('kode'=>'09050191','tahun'=>'2017'),
                        array('kode'=>'09050192','tahun'=>'2017')
                        );
$KRS=array('nama'=>'Agus','alamat'=>'Solo','krs'=>$MK);
$Data=json_encode($KRS);
echo $Data;
?>
```

Disimpan di htdocs/sister.1/

```
<?php
$Data=file_get_contents('http://localhost/sister.1/jsonMultidimensiServer.php');
$DataHasil=json_decode($Data);
echo '<pre>';
print_r($DataHasil);
echo '</pre>';
echo $DataHasil->nama;
echo '<br/>';
echo     'Kode Matkul 1: '.$DataHasil->krs[0]->kode;
echo '<br/>';
echo    'Kode Matkul 2: '.$DataHasil->krs[1]->kode;
// Anda bisa melakukan Looping utk membaca record KRS 0 sampai ke n
?>
```

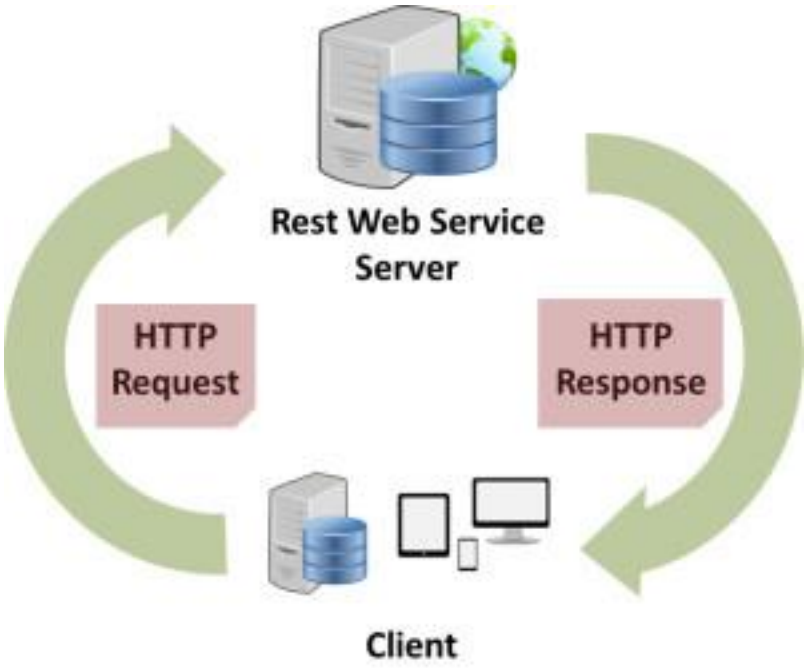
## **Tugas**,

Buatlah database Mahasiswa yang terdiri dari 2 tabel yaitu mahasiswa dan krs dimana untuk tabel mahasiswa terdiri atas 4 field yaitu nim, nama, alamat dan tanggal lahir dan tabel krs terdiri atas 5 field, yaitu id,nim,tahun,semester dan nilai. Selanjutnya

1. buatlah REST API dalam format JSON untuk menampilkan request data nim,nama dan KRS yang diambil pada periode tertentu.
2. buatlah aplikasi berbasis PHP yang berisi text input yang ketika diisikan nim, tahun dan semester akan menampilkan nim, nama dan KRS yang diambil dengan mengakses REST API pada angka 1 di atas.

# Mengenal REST API - XML

**Pendahuluan**




Gambar 1. Rest Web Service

**REST** (*Representional State Transfer*) adalah suatu arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data dan metode ini sering diterapkan dalam pengembangan aplikasi, sedangkan **API** singkatan dari *Application Programming Interface* yaitu sekumpulan perintah, fungsi, serta protokol yang dapat digunakan oleh programmer saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan programmer untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi.

**XML (Extensible Markup Language)** adalah bahasa markup untuk keperluan umum yang disarankan oleh W3C untuk membuat dokumen markup keperluan pertukaran data antar sistem yang beraneka ragam. XML didesain untuk mempu menyimpan data secara ringkas dan mudah diatur. XML menyediakan suatu cara terstandarisasi namun bisa dimodifikasi untuk menggambarkan isi dari dokumen. Dalam pengertian yang sederhana, sebuah dokumen XML hanyalah sebuah fi le teks biasa yang berisikan berbagai tag yang didefi nisikan sendiri oleh pembuat dokumen XML tersebut. Sesuai dengan namanya, eXtensible Markup Language, sebuah dokumen XML adalah sebuah dokumen dengan markup, sama seperti halnya dengan HTM

XML memiliki tiga tipe file :

a. XML, merupakan standar format dari struktur berkas (file).
b. XSL, merupakan standar untuk memodifikasi data yang diimpor atau diekspor.
c. XSD, merupakan standar yang mendefinisikan struktur database dalam XML.

**Keunggulan XML bisa diringkas sebagai berikut :**

1. Pintar (Intelligence). XML dapat menangani berbagai tingkat (level) kompleksitas.
2. Dapat beradaptasi. Dapat mengadaptasi untuk membuat bahasa sendiri. Seperti Microsoft membuat bahasa
3. MSXML atau Macromedia mengembangkan MXML.
4. Mudah pemeliharaannya.
5. Sederhana. XML lebih sederhana.
6. Mudah dipindah-pindahkan (Portability). XML mempunyai kemudahan perpindahan (portabilitas) yang lebih bagus.

Contoh XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tujuan>email@tujuan.com</tujuan>
<dari>asal@asal.com</dari>
<subyek>Contoh Subyek</subyek>
<body>Isi Konten pesan yg dikirim</body>
```

## Bahan Praktikum

Komunikasi Data Menggunakan JSON

1. **Data Array Sederhana**

   **File jsonMahasiswaServer.php**

   ```
   <?php

   $DataMahasiswa=array();
   $DataMahasiswa=array('nim'=>'M0516001','nama'=>' Iwan
   Subagya','alamat'=>'Surakarta','tempatLahir'=>'Surakarta','tglLahir'=>'1997-01-01');
   echo '<mhs>';
   ```

```
        echo  '<nim>'.$DataMahasiswa['nim'].' </nim>';
        echo '<nama>' .$DataMahasiswa['nama'].' </nama>';;
        echo '<alamat>'.$DataMahasiswa['alamat'].'</alamat>';
        echo '<tglLahir>' .$DataMahasiswa['tglLahir].'</tglLahir>';
echo '</mhs>';
?>
```

Disimpan di htdocs/sister.2/xmlMahasiswaServer.php
File ini merupakan file yang digunakan sebagai sumber data. Selanjutnya akan dibuat file untuk memanggil sebagai aplikasi client.

File **xmlMahasiswaClient.php**

```
<?php
$Data=simplexml_load_file ('http://localhost/sister.2/xmlMahasiswaServer.php');
echo 'Nama:'.$Data->mhs->nama;
echo '<br/>';
echo 'Tanggal Lahir: '.$Data->mhs->tglLahir;

?>
```

Disimpan di folder htdocs/sister.1

2. **Data Array Multidimensi dengan Function Array to XML**

File **jsonMultidimensiServer.php**

```
<?php
        $KRS=array();
        $MK=array(
                        array('kode'=>'09050191','tahun'=>'2017'),
                        array('kode'=>'09050192','tahun'=>'2017')
                        );
$KRS=array('nama'=>'Agus','alamat'=>'Solo','krs'=>$MK);
                function array_to_xml( $data, &$xml_data ) {
                   foreach( $data as $key => $value ) {
                      if( is_numeric($key) ){
                         $key = 'item'.$key; //dealing with <0/>..<n/> issues
                      }
                      if( is_array($value) ) {
                         $subnode = $xml_data->addChild($key);
                         array_to_xml($value, $subnode);
                      } else {
                         $xml_data->addChild("$key",htmlspecialchars("$value"));
                      }
                   }
                }
echo $xml = new SimpleXMLElement('<mhs/>');
array_to_xml($DataMahasiswa,$xml);
print $xml->asXML();
?>
```

Disimpan di htdocs/sister.2/

```
<?php
```

```
$Data='http://localhost/sister.2/xmlMultidimensiServer.php';
$DataHasil=simplexml_load_file($Data);
echo '<pre>';
print_r($DataHasil);
echo '</pre>';
echo $DataHasil->nama;
echo '<br/>';
echo     'Kode Matkul 1: '.$DataHasil->krs->item0->kode;
echo '<br/>';
echo    'Kode Matkul 2: '.$DataHasil->krs->item1->kode;
// Anda bisa melakukan Looping utk membaca record KRS 0 sampai ke n
?>
```

**Tugas**,

Buatlah database Mahasiswa yang terdiri dari 2 tabel yaitu kuitansi dan detail_penjualan dimana untuk tabel kuitansi terdiri atas 4 field yaitu id_kuitansi, no_kuitansi, tanggal_transaksi, id_user dan tabel detail_penjualan terdiri atas 5 field, yaitu id,id_kuitansi,id_barang,jumlah_barang, diskon dan harga_satuan. Selanjutnya

1. buatlah REST API dalam format JSON untuk menampilkan request data kuitansi dan detail penjualan berdasarkan nomor kuitansi.
2. buatlah aplikasi berbasis PHP yang berisi text input yang ketika diisikan nomor kuitansi akan menampilkan tanggal_transaksi, id_barang, jumlah_barang,harga_satuan dan diskon yang diambil dengan mengakses REST API pada angka 1 di atas.