

Pemrograman Jaringan 10

Komunikasi Antar Obyek

Komunikasi Antar Obyek

- Dalam pemrograman berbasis obyek, antar obyek harus saling berkomunikasi
- Komunikasi ini dapat berupa **pemanggilan method** dari obyek lain ataupun secara langsung mengakses **atribut** dari obyek lain.
- Agar dapat terjadi komunikasi, setiap obyek harus dibuat dari **satu interface**.
 - Jadi setiap obyek harus memiliki **definisi interface**.
- Pemanggilan method yang terjadi antar obyek pada proses yang berbeda disebut sebagai *remote method invocation*.
- Sedangkan pemanggilan method antar obyek dalam satu proses yang sama disebut *local method invocation*. Suatu obyek yang dapat menerima *remote invocation* disebut *remote object*.

Komunikasi Antar Obyek

- Suatu *remote object* dapat dipanggil methodnya dari obyek pada proses yang berbeda melalui sebuah ***remote object references***.
- *Remote object references* merupakan **ID** yang digunakan pada suatu remote obyek tertentu, baik pada satu mesin ataupun pada mesin yang berbeda.
- Pada ID Remote references terdapat informasi alamat **host** dimana remote obyek yang ditunjuk berjalan
- Suatu remote obyek memiliki **remote interface** yang mendefinisikan **method** mana yang dapat dipanggil secara remote.

Serializable

- Pada pemrograman socket biasanya yang dikirim adalah **data stream**.
- Nah, bagaimana jika yang dikirim adalah **suatu obyek**?
 - Hal ini dapat dilakukan dengan menggunakan serialisasi obyek.
- *Object Serialization* adalah teknik dimana suatu program dapat menyimpan **status obyek** ke dalam sebuah **file** dan kemudian dapat dipanggil kembali dari file ke memori atau dikirim melalui jaringan.
- Serialization flattens objects into an ordered, or **serialized stream** of bytes
- The ordered stream of bytes can then be read at a later time, or in another environment, to recreate the original objects.

Serializable

- Jika sebuah obyek ingin diserialisasi, maka obyek itu harus mengimplementasikan **java.io.Serializable** atau **java.io.Externalizable**
- Untuk menuliskan obyek yang terserialisasi ke file dibutuhkan I/O stream khusus, yaitu menggunakan ***ObjectOutputStream*** yang merupakan subclass dari ***FilterOutputStream***.
- Java serialization **does not** occur for transient or static fields

Contoh Pegawai

```
import java.io.*;

public class Pegawai implements Serializable {
    private String nama;
    private int umur;
    private int gaji;

    public Pegawai(String nama, int umur, int gaji) {
        this.nama = nama;
        this.umur = umur;
        this.gaji = gaji;
    }

    public void print() {
        System.out.println("Data untuk " + this.nama);
        System.out.println("nama " + this.nama);
        System.out.println("umur " + this.umur);
        System.out.println("gaji " + this.gaji);
    }
}
```

Contoh SimpanPegawai

```
import java.io.*;

public class SimpanPegawai {
    public static void main(String[] args) {
        Pegawai aaa = new Pegawai("aaa", 28, 100);
        Pegawai bbb = new Pegawai("bbb", 30, 150);

        try {
            FileOutputStream fos = new FileOutputStream("db");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(aaa);
            oos.writeObject(bbb);
            oos.flush();
        }
        catch (IOException e) {
            System.out.print("error " + e);
            System.exit(1);
        }
    }
}
```

Penjelasan

- Berarti program SimpanPegawai akan menyimpan 2 obyek pegawai yaitu “aaa” dan “bbb” ke dalam file bernama “db”.
- Sedangkan untuk pembacaan file yang berisi obyek juga harus dilakukan secara berurutan, yaitu “aaa” dulu baru “bbb”.
- Karena pembacaan dengan menggunakan **readObject()** yang mengembalikan Object, maka harus dilakukan **casting** sesuai dengan tipe Objectnya.

Contoh: BacaPegawai

```
import java.io.*;

public class BacaPegawai {
    public static void main(String[] args) {
        try {
            FileInputStream fis = new FileInputStream("db");
            ObjectInputStream ois = new ObjectInputStream(fis);

            Pegawai aaa = (Pegawai) ois.readObject();
            Pegawai bbb = (Pegawai) ois.readObject();

            aaa.print();
            bbb.print();
        }
        catch (IOException e) {
            System.exit(1);
        }
        catch (Exception e) {
            System.exit(1);
        }
    }
}
```

```
Data untuk aaa
nama aaa
umur 28
gaji 100
Data untuk bbb
nama bbb
umur 30
gaji 150
```

Serialisasi dgn Vector

```
import java.io.*;
import java.util.*;

public class VectorSerial {

    public static void main(String args[]) throws IOException, ClassNotFoundException {

        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("filepeg.dat"));
        Vector<Pegawai> staffout = new Vector<Pegawai>();
        Vector<Pegawai> staffin = new Vector<Pegawai>();

        Pegawai[] staff = {new Pegawai("anton",20,1000), new Pegawai("yuan",21,1500), new Pegawai("Mahas",22,2000) };

        for (int i = 0; i<staff.length; i++)
            staffout.add(staff[i]);
        oos.writeObject(staffout);
        oos.close();

        ObjectInputStream ois = new ObjectInputStream(new FileInputStream("filepeg.dat"));
        try {
            staffin = (Vector<Pegawai>)ois.readObject();
            for(Pegawai p:staffin){
                p.print();
            }
        }
        catch (EOFException ex) {
            System.out.println (ex.getMessage());
            ois.close();
        }

        System.console().readLine();
    }
}
```

```
Data untuk anton
nama anton
umur 20
gaji 1000
Data untuk yuan
nama yuan
umur 21
gaji 1500
Data untuk Mahas
nama Mahas
umur 22
gaji 2000
```

Serialisasi dengan ArrayList

```
import java.io.FileInputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.util.ArrayList;
import java.util.List;

public class GetPegawaiDetails {

    public static void main(String[] args) {
        String filename = "pegawai.txt";
        List pDetails = null;
        FileInputStream fis = null;
        ObjectInputStream in = null;
        try {
            fis = new FileInputStream(filename);
            in = new ObjectInputStream(fis);
            pDetails = (ArrayList) in.readObject();
            in.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }
        // print out the size
        System.out.println("Pegawai Details Size: " + pDetails.size());
        for (int i = 0; i < pDetails.size(); i++) {
            Pegawai p = (Pegawai) pDetails.get(i);
            p.print();
        }
        System.console().readLine();
    }
}
```

```
Pegawai Details Size: 3
Data untuk Hemanth
nama Hemanth
umur 50
gaji 1000
Data untuk Bob
nama Bob
umur 52
gaji 2000
Data untuk Richa
nama Richa
umur 40
gaji 3000
```

Transient

- Dalam contoh-contoh diatas, **semua atribut** dari kelas Pegawai secara **otomatis** bisa dibaca karena bersifat **serializable**.
- Kita bisa **membatasi** hak akses terhadap atribut tertentu saja yang bersifat serializable.
- Hal ini dapat dilakukan dengan menggunakan keyword ***transient***.
- Dengan keyword itu maka atribut tersebut tidak ikut “terbawa” untuk dikirimkan dalam deretan byte melalui I/O Stream.
 - the transient variable **is not part** of the persistent state of the object and thus never saved during serialization.
- You can use the transient keyword to describe *temporary variables*, or variables that contain local information, such as a *process ID* or a *time lapse*.

Pegawai yang Transient

```
import java.io.*;

public class Pegawai implements Serializable {
    private String nama;
    private transient int umur;
    private int gaji;

    public Pegawai(String nama, int umur, int gaji) {
        this.nama = nama;
        this.umur = umur;
        this.gaji = gaji;
    }

    public void print() {
        System.out.println("Data untuk " + this.nama);
        System.out.println("nama " + this.nama);
        System.out.println("umur " + this.umur);
        System.out.println("gaji " + this.gaji);
    }
}
```

```
Data untuk aaa
nama aaa
umur 0
gaji 100
Data untuk bbb
nama bbb
umur 0
gaji 150
```

Penjelasan

- Atribut “**umur**” akan tetap ditampilkan tapi nilainya 0 -> hal ini karena keyword transient
- Kita tetap dapat melakukan manipulasi atribut yang bersifat transient, agar tetap dapat disimpan dan dibaca hasilnya.
- Untuk itu kita harus melakukan *overriding* method **readObject()** dan **writeObject()**.
- Pada method **writeObject()** terdapat **stream.defaultWriteObject()** yang akan menuliskan ke Stream seperti defaultnya, yaitu dimana atribut transient tidak ditulis.
- Sedangkan untuk menuliskan yang transient kita gunakan **stream.writeObject(b);**
- Demikian pula dengan method **readObject()** terdapat **stream.defaultReadObject()** yang akan membaca Stream seperti biasa, dimana atribut yang transient tidak terbaca.
- Sedangkan untuk membaca yang transient kita gunakan **stream.readObject(b);**

SerialCtl

```
import java.io.*;

public class SerialCtl implements Serializable {
    private String a;
    private transient String b;

    public SerialCtl(String aa, String bb) {
        a = "Not Transient: " + aa;
        b = "Transient: " + ((bb == null) ? "(n/a)" : bb);
    }

    public String toString() {
        return a + "\n" + ((b == null) ? "(n/a)" : b);
    }

    private void writeObject(ObjectOutputStream stream)
        throws IOException {
        stream.defaultWriteObject();
        stream.writeObject(b);
    }

    private void readObject(ObjectInputStream stream)
        throws IOException, ClassNotFoundException {
        stream.defaultReadObject();
        b = (String)stream.readObject();
    }
}
```

SerialCtlMain

```
import java.io.*;

public class SerialCtlMain {
    public static void main(String[] args)
        throws IOException, ClassNotFoundException
    {
        SerialCtl sc = new SerialCtl("Test1", "Test2");
        System.out.println("Before:\n" + sc);

        FileOutputStream fos =
            new FileOutputStream("dbExCtl");
        ObjectOutputStream oos =
            new ObjectOutputStream(fos);
        oos.writeObject(sc);
        oos.flush();
        oos.close();

        FileInputStream fis = new FileInputStream("dbExCtl");
        ObjectInputStream ois = new ObjectInputStream(fis);

        SerialCtl sc2 = (SerialCtl)ois.readObject();
        System.out.println("After:\n" + sc2);
    }
}
```

Before:

Not Transient: Test1

Transient: Test2

After:

Not Transient: Test1

Transient: Test2

Override

```
private void writeObject(ObjectOutputStream stream)
    throws IOException {
    stream.defaultWriteObject();
    //stream.writeObject(b);
}
```

Before:

Not Transient: Test1

Transient: Test2

After:

Not Transient: Test1

(n/a)

- Selain menggunakan overriding method `writeObject()` dan `readObject()`, kita dapat membuat sebuah kelas yang mengimplementasikan **Externalizable**.
- **Externalizable** adalah subclass dari **Serializable** yang memiliki dua method yang perlu dioverride yaitu **writeExternal** dan **readExternal**.

PegawaiEx

```
import java.io.*;

public class PegawaiEx implements Externalizable {
    private String nama;
    private int umur;
    private int gaji;

    public PegawaiEx() {
    }

    public PegawaiEx(String nama, int umur, int gaji) {
        this.nama = nama;
        this.umur = umur;
        this.gaji = gaji;
    }

    public void writeExternal(ObjectOutput objout)
        throws IOException {
        objout.writeObject("Nama : " + this.nama);
        objout.writeInt(this.umur);
        objout.writeInt(this.gaji);
    }

    public void readExternal(ObjectInput objin)
        throws IOException, ClassNotFoundException {
        this.nama = (String)objin.readObject();
        this.umur = objin.readInt();
        this.gaji = objin.readInt();
    }

    public String toString() {
        return "Data untuk " + this.nama + "\n" +
            this.nama + "\n" +
            "umur " + this.umur + "\n" +
            "gaji " + this.gaji;
    }
}
```

SimpanPegawaiEx

```
import java.io.*;

public class SimpanPegawaiEx {
    public static void main(String[] args) {
        PegawaiEx aaa = new PegawaiEx("aaa", 28, 100);
        PegawaiEx bbb = new PegawaiEx("bbb", 30, 150);

        System.out.println(aaa);
        System.out.println(bbb);

        try {
            FileOutputStream fos = new FileOutputStream("dbEx");
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(aaa);
            oos.writeObject(bbb);
            oos.flush();
        }
        catch (IOException e) {
            System.out.print("error " + e);
            System.exit(1);
        }
    }
}
```

BacaPegawaiEx

```
import java.io.*;

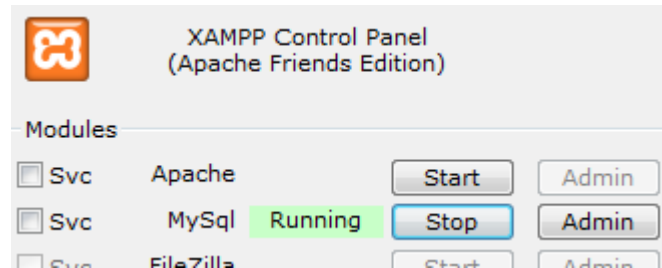
public class BacaPegawaiEx {
    public static void main(String[] args)
        throws IOException, ClassNotFoundException {
        FileInputStream fis = new FileInputStream("dbEx");
        ObjectInputStream ois = new ObjectInputStream(fis);

        Object aaa = (PegawaiEx) ois.readObject();
        Object bbb = (PegawaiEx) ois.readObject();

        System.out.println(aaa);
        System.out.println(bbb);
    }
}
```

```
Data untuk Nama : aaa
Nama : aaa
umur 28
gaji 100
Data untuk Nama : bbb
Nama : bbb
umur 30
gaji 150
```

Serialisasi pada Database



#	Name	Datatype	Length/Set	Unsign...	Allow N...	Default
1	name	BLOB			<input checked="" type="checkbox"/>	No default
2	viewname	VARCHAR	30		<input checked="" type="checkbox"/>	NULL

Columns:

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Default
1	name	BLOB			<input checked="" type="checkbox"/>	No default
2	viewname	VARCHAR	30		<input checked="" type="checkbox"/>	NULL

```
create table SerialTest(  
name BLOB, viewname VARCHAR(30)  
);
```

```
copy mysql JDBC driver ke JRE\lib\ext
```

Serialisasi pada Database

```
//query:
//create table SerialTest(
//name BLOB,
//viewname VARCHAR2(30)
//);
-//copy mysql ke jre\lib

} import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
- import java.sql.Statement;

} public class PegawaiPersistDB {

    static String userid = "root", password = "";
    static String url = "jdbc:mysql://localhost:3306/test";
    static int count = 0;
    static Connection con = null;
} public static void main(String[] args) {
    Connection con = getMySQLJDBCConnection();
    Pegawai person1 = new Pegawai("Hemanth", 50, 1000);
    Pegawai person2 = new Pegawai("Bob", 52, 2000);
    Pegawai person3 = new Pegawai("Richa", 40, 3000);
    PreparedStatement ps;
```

```

try {
    ps = con.prepareStatement("INSERT INTO SerialTest VALUES (?, ?)");
    write(person1, ps);
    ps.execute();
    write(person2, ps);
    ps.execute();
    write(person3, ps);
    ps.execute();
    ps.close();
    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery("SELECT * FROM SerialTest");
    while (rs.next()) {
        Object obj = read(rs, "Name");
        Pegawai p = (Pegawai) obj;
        p.print();
    }
    rs.close();
    st.close();
} catch (Exception e) {
}
}

public static void write(Object obj, PreparedStatement ps)
    throws SQLException, IOException {
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    ObjectOutputStream oout = new ObjectOutputStream(baos);
    oout.writeObject(obj);
    oout.close();
    ps.setBytes(1, baos.toByteArray());
    ps.setInt(2, ++count);
}

```

```

public static Object read(ResultSet rs, String column)
    throws SQLException, IOException, ClassNotFoundException {
    byte[] buf = rs.getBytes(column);
    if (buf != null) {
        ObjectInputStream objectIn = new ObjectInputStream(
            new ByteArrayInputStream(buf));
        return objectIn.readObject();
    }
    return null;
}
public static Connection getMySQLJDBCConnection() {
    try {
        Class.forName("org.gjt.mm.mysql.Driver");
    } catch (java.lang.ClassNotFoundException e) {
        System.err.print("ClassNotFoundException: ");
        System.err.println(e.getMessage());
    }
    try {
        con = DriverManager.getConnection(url, userid, password);
    } catch (SQLException ex) {
        System.err.println("SQLException: " + ex.getMessage());
    }
    return con;
}
}

```


Hasil

```
E:\Documents\Dosen\progjar\2008\serial>java PegawaiPersistDB2
Data untuk Richa
nama Richa
umur 40
gaji 3000
Data untuk Hemanth
nama Hemanth
umur 50
gaji 1000
Data untuk Bob
nama Bob
umur 52
gaji 2000
```

name	viewname
0xACED00057372000750656761776169BE9ADF7FB7A63A5C02000349000467...	3
0xACED00057372000750656761776169BE9ADF7FB7A63A5C02000349000467...	1
0xACED00057372000750656761776169BE9ADF7FB7A63A5C02000349000467...	2

Serialisasi dengan **Socket**

- Buat kelas PegawaiServer
- Buat kelas PegawaiClient
- PegawaiServer mengirim data serialisasi obyek ke PegawaiClient
- PegawaiClient membaca

```

import java.io.*;
import java.net.*;
public class PegawaiServer {
    public static void main(String[] args) {
        ServerSocket ser = null;
        Socket soc = null;
        Object pegawaiInput = null;
        try {
            ser = new ServerSocket(4545);
            System.out.println("Server siap pada port 4545");
            soc = ser.accept();

            ObjectInput ois = new
                ObjectInputStream(soc.getInputStream());
            ObjectOutput ous = new
                ObjectOutputStream(soc.getOutputStream());

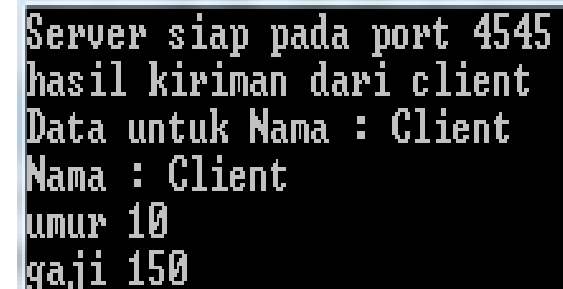
            pegawaiInput = (PegawaiEx) ois.readObject();

            System.out.println("hasil kiriman dari client");
            System.out.println(pegawaiInput);

            String res = new
                String("Data Pegawai telah diterima!");
            ous.writeObject(res);
            ous.flush();

            ous.close();
            ois.close();
        }
        catch (Exception e) {
            e.printStackTrace();
            System.out.println(e.getMessage());
        }
    }
}

```



```

Server siap pada port 4545
hasil kiriman dari client
Data untuk Nama : Client
Nama : Client
umur 10
gaji 150

```

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

public class PegawaiClient {
    public static void main(String[] args) {
        Socket soc = null;
        PegawaiEx pegawaiInput = null;

        try {
            soc = new Socket("localhost", 4545);

            ObjectOutputStream ous = new
                ObjectOutputStream(soc.getOutputStream());
            ObjectInput ois = new
                ObjectInputStream(soc.getInputStream());

            pegawaiInput = new PegawaiEx("Client", 10, 150);

            ous.writeObject(pegawaiInput);
            ous.flush();

            String s = (String) ois.readObject();

            System.out.println(s);

            ous.close();
            ois.close();
        }
        catch (Exception e) {
            e.printStackTrace();
            System.out.println(e.getMessage());
        }
    }
}
```

```
E:\Documents\Dosen\progjar\2008\serial\Externalize>java PegawaiClient
Data Pegawai telah diterima!
```

Buat program

- Buat program PegawaiServer
- Buat program PegawaiClient
- Dengan menggunakan connection socket oriented
- Server mengirim **file menggunakan vector** yang berisi pegawai terserialisasi
- Client menerimanya dan menampilkan isinya!

Next

- RMI