ArtiVisi's Skill Enhancement Series

Lab Module

Java Advanced

ver 1.0.1 - 0903

Authors :
Endy Muhardin
I Putu Artha Kristiwan
Nurhasyim
Nursapta

ArtiVisi Intermedia
http://www.artivisi.com
contact :
support@artivisi.com

About the document

Copyright

Disclaimer

No liability for the contents of this documents can be accepted. Use the concepts, examples and other content at your own risk. As this is a new edition of this document, there may be errors and inaccuracies, that may of course be damaging to your system. Proceed with caution, and although this is highly unlikely, the author(s) do not take any responsibility for that.
You are strongly recommended to take a backup of your system before major installation and backups at regular intervals.

Version

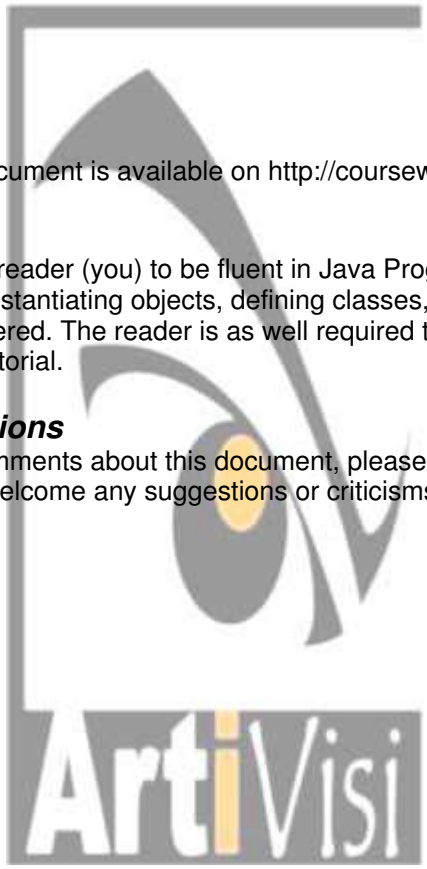
Revision : 1.0.1
Date : September 2003
The latest version of this document is available on http://courseware.artivisi.com

Knowledge Required

This document assume the reader (you) to be fluent in Java Programming Fundamentals. Basic capabilities such as instantiating objects, defining classes, methods, attributes, and inner classes must be mastered. The reader is as well required to be able to use Java API documentation and Java Tutorial.

Feedback and corrections

If you have questions or comments about this document, please feel free to mail us at support@artivisi.com. We welcome any suggestions or criticisms. Thanks.



SESSION I
Advance Swing

Subject :
Advance Swing Component

Objective :

- Understanding Progress Bar
- Understanding Progress Bar Input Stream

Preface :

- Explain what do you know about Progress Bar ?
- Explain what do you know about Progress Bar Input Stream

Exercises :

1. Create a program that use a Progress Bar component, this program will show the percentage progress of looping process from 0 to 1000. Follow this instruction bellow to do this exercise :
   - Create user interface (see Figure 1), add all component that are needed such as JProgressBar, JTextArea, JButton. Use the layout manager that are needed too.
   - When button **Start** pressed looping process will start and so the Progress Bar. When the button **Start** pressed this will invoke the Thread (see next step bellow to create thread)
   - Create a Thread where inside of this, make a looping process from 0 to 1000.

   To create a Thread :
   - implements interface ***Runnable*** on your class definition.
   - Override method ***public void run()*** inside your class.
   - Set the delay of this Thread using method ***sleep(delay_time)*** .
   - Use Exception handling method to cover system error or runtime error in this case **InterruptedException**

   Inside looping process when the value increase one by one, set the Progress Bar value using method ***setValue(increasing_value)*** and append increasing value into text area.

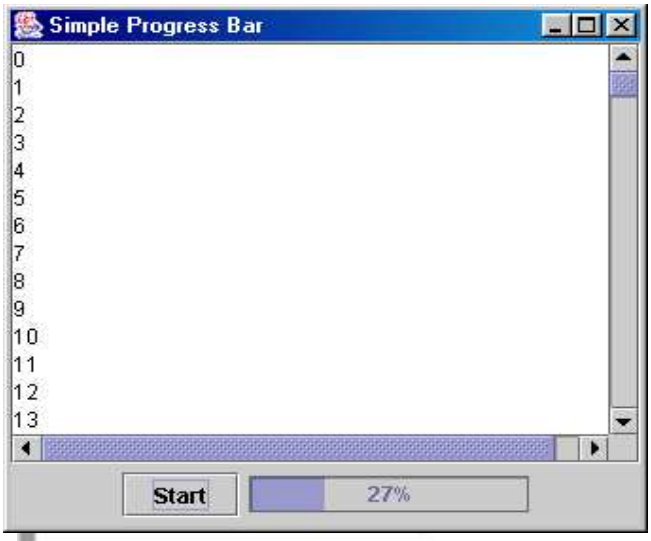


Figure 1

Do It Your Self :

1. Create program that able to read a text file, see this spesification bellow and Figure 2 and Figure 3:
   - choose the text file that will be read with, press the **Open** button
   - when button **Open** is pressed the file chooser dialog will appear (see Figure 2). File choosen will appear int text field near the **Open** button, complete with the path.
   - Press button **Read**, the program will begin read the containt of file text that choosen above. Every read one character from file text, the character will appear inside the text area and the **Progress Bar** will increase too.
2. Follow this instruction bellow :
   - Create the user interface, see Figure 3
   - Add listener to button **Open.** And write this code bellow inside of that :

```
int
returnVal=fileChooser.showOpenDialog(ReadFile.
this);
if(returnVal == JFileChooser.APPROVE_OPTION) {
     file=fileChooser.getSelectedFile();
     String path=file.getPath();
     fileName.setText(path);
}
```

This program will appear the file chooser dialog.

-

- Inside that :
- read the text file using I/O
- every time read a character from text file increase the Progress Bar.
- Every time read a character from text file append the character into text area

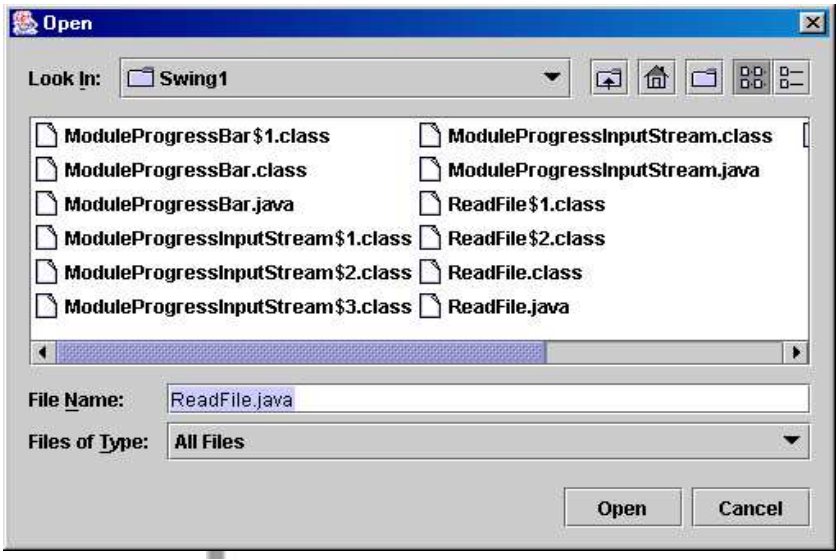


Figure 2

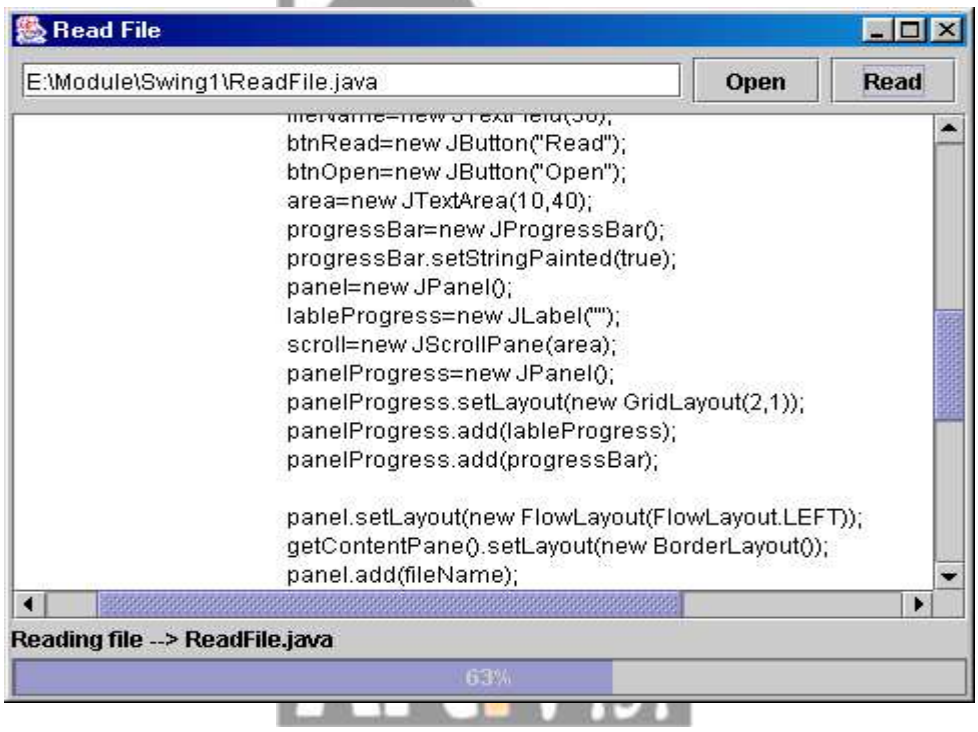


Figure 3

SESSION II
ADVANCE SWING

Subject :

Concepts of Model View Controller and Jtable

Objective :

- Understanding JTable component
- Understanding AbstractTableModel class

Preface :

1. Explain what do you know about JTable component ?
2. Explaint what do you know about AbstractTableModel class ?

Exercises :

2. Create a simple program that use JTable component (see Figure 1). This program will show the data in table format. Follow this instruction bellow :
   - Create a user interface (see Figure 1), add all component that are needed such as JTable, JscrollPane. Add JTable component to JScrollPane.
   - In your class define two kind of array as follow :

```
- Object[][] data={new Integer(1),"Rajes
  Khan","NewDelhi"},{…,…,…}}
- String[] header={"Enroll Number", "Student Name",
  "Address"}
```

   Column header will get from header array and data of the table will git from data array. When the data and the column header will get from both array ?. The answer is when you create a table (instance the JTable class)

Figure 1

3.
   - Create a user interface (see Figure 1), add all component that are needed such as JTable, JscrollPane. Add JTable component to JScrollPane.
   - Define an inner class that extends AbsractTableModel class like bellow

```
class MyTableModel extends AbstractTableModel{
      public String getColumnName(int column){
            return header.get(column).toString();
      }

      public int getColumnCount(){
      return header.size();
      }

      public int getRowCount(){
            return data.size();
      }

      public Object getValueAt(int row, int column){
            Vector vec=(Vector)data.get(row);
            return vec.get(column);
      }
}
```

-

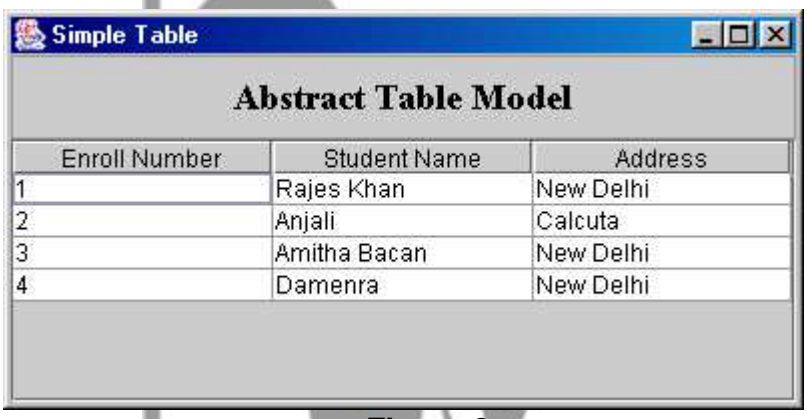


Figure 2

Do It Your Self :

1. Create a program that have availablelity as you see in Figure 3 bellow.
2. Program Specification :

- when the user input the data into text field (Number and Name), and the button **Add** is pressed, then that data will be added to the table
- when the button **Clear** is pressed then all the data in table will be removed

3. Instruction

- Use AbstractTableModel class to construct thet program
- Use method ***fireTableRowsInserted(int rowSize,int colSize)*** to update the table.

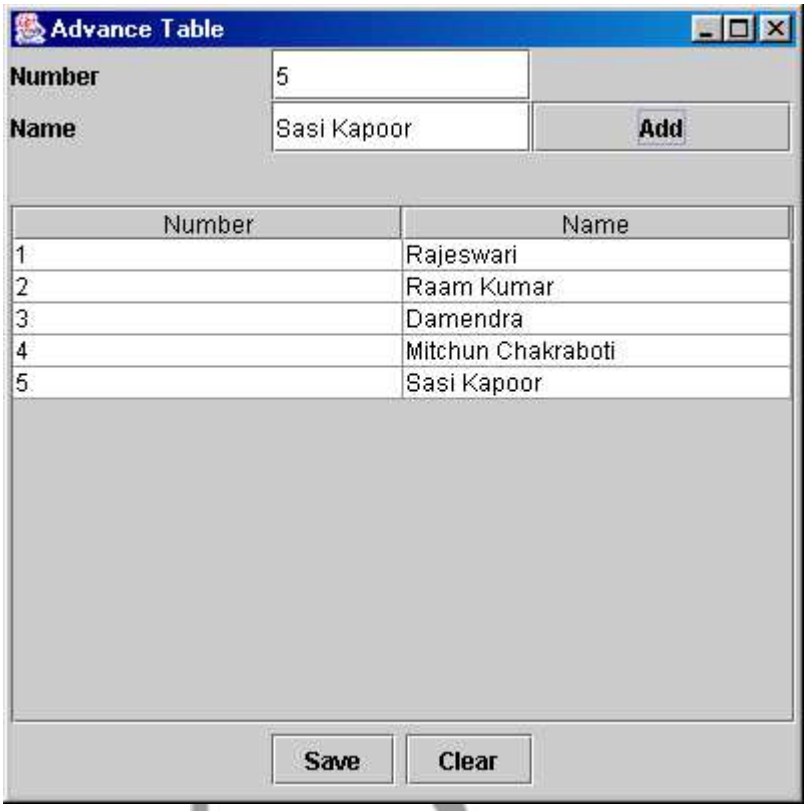


Figure 3

SESSION III
Java Database Connectivity

Subject :
JDBC Driver

Objective :
- Understanding JDBC Driver
- Understanding Native Driver
- SQL Server for JDBC Driver

Exercises :
1. Install SQL Sever for JDBC Driver.

Instruction :
- copy file mssqlserver.jar, msutil.jar, msbase.jar into java home directory, folder jre, lib, ext
  exp : c:\j2sdk1.4.0_01\jre\lib\ext\mssqlserver.jar
  c:\j2sdk1.4.0_01\jre\lib\ext\msutil.jar
  c:\j2sdk1.4.0_01\jre\lib\ext\msbase.jar
- This simple program show to you how to use the driver

```
import java.sql.*;
class TestingDriver{
  public static void main(String[] args){
    String dbDrv =
         "com.microsoft.jdbc.sqlserver.SQLServerDriver";
    String dbUrl = "jdbc:microsoft:sqlserver://server;";
    dbUrl +="DatabaseName=pubs;User=sa;Password=""";
    try{
      Class.forName(dbDrv);
      Connection c = DriverManager.getConnection(dbUrl);
    }catch(ClassNotFoundException ce){
       System.err.println(ce);
    }
  }
}
```

- Create program to test the driver that istalled. This program is a simple program that can display emp_id, fname, and lname from table employee in database pubs (see Figure 1).
- To get data from table :
- create a Statement

```
Statement stm=connection.createStatement();
```

- execute the statement with specified query

```
ResultSet result =
       stm.executeQuery("select emp_id, fname, lname from
employee");
```

- looping the ResultSet to display data

Figure 1

Do It Your Self :

1. Create a program to make a table in a database (see Figure 2)
2. Program specification :

- the program can make a table into database
- user should insert data that require to make a table, as follow :
- Table Name, Field Name, Data Type, Konstraint (PRIMARY KEY, NOT NULL, and so on)

3. Instruction :

- when button **Add** is pressed, then Field Name, Data Type, Konstraint will be save in a Vector. Repeat this step to complete your Field Name that you need.
- When button **Create** is pressed, then program will create a table into database.

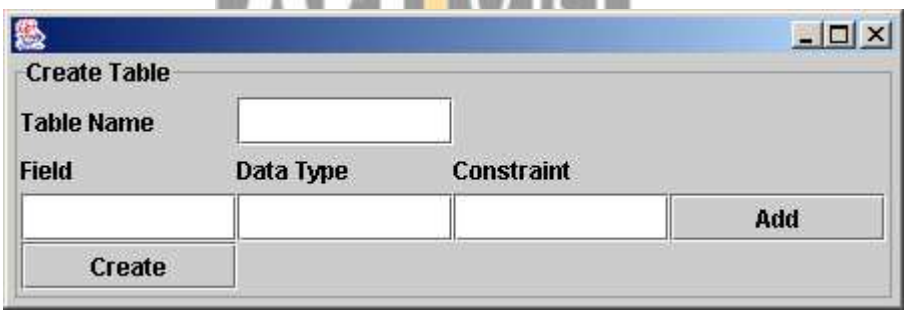


Figure 2

SESSION IV
Java Database Connectivity

Subject :
Using Native Driver

Objectives :
- Capable of using advanced resultset
- Capable of using PreparedStatements

Exercises :
1. Create GUI as shown in figure 4.1

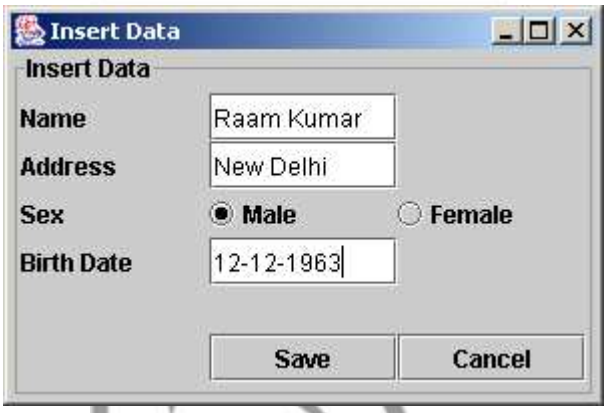


Figure 4.1
Insert Form

2. Create class Person with the following attribute :
   - Full Name
   - Address
   - Sex
   - BirthDate

   along with the accessor and mutator methods.

3. Create an interface PersonData with the following method:
   public boolean create(String name);
   public boolean update(Person oldData, Person newData);

4. Create a DAO class implementing PersonData with the following steps:
   - Declare class XxxPersonData [replace Xxx with database engine you are using]
   - Declare object variable ResultSet and Connection
   - Add connection initialization codes in the constructor, passing the return value to the object variable.
   - Implement public Person create (Person p) method using updatable resultset
   - Add ResultSet initialization code in the method.
   [hints : use ResultSet.TYPE_SCROLL_SENSITIVE and ResultSet.CONCUR_UPDATABLE options]
   - Add code to insert new row using the specified object in the parameter
   [hints : use moveToInsertRow, updateXxx, and insertRow method of the resultset]

5. Check your result

Do it Yourself :

Implement public boolean update (Person oldData, Person newData) method as per the explanation above, optionally use PreparedStatement for the query.

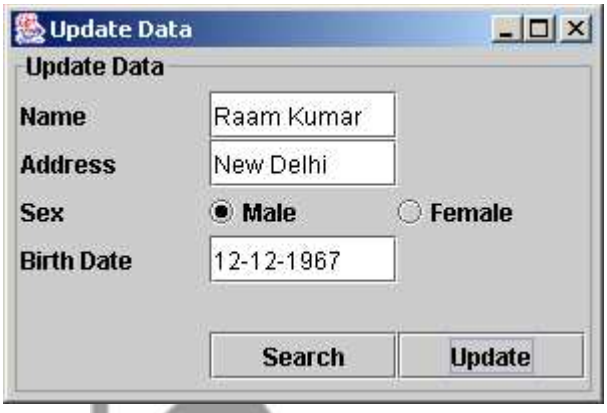


Figure 4.2
Update Form

SESSION V
Java Database Connectivity

Subject :
JDBC Batch Update

Objective :
- Understanding Batch Update

Exercises :

1. Create simple program to test batch update, this program will insert data into table Personal in database. But before data insert into table Personal, insert command first add into batch update.
2. **Instruction :**
   - Make connection to database using the following code :

```
Connection con =
  DriverManager.getConnection(Url,"username","password");
```

   - Make a statement using the following code

```
Statement stmt =
     con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                         ResultSet.CONCUR_UPDATABLE);
```

   - Add command to batch update

```
stmt.addBatch("INSERT INTO Personal VALUES('Amaretto',
'Italia',1,'12-12-1963')");

stmt.addBatch("INSERT INTO Personal VALUES('Hazelnut',
'Italia',1,'12-12-1963')");

stmt.addBatch("INSERT INTO Personal VALUES('Amaretto
Decafi','Italia',1,'12-12-1963')");

stmt.addBatch("INSERT INTO Personal VALUES('Hazelnut
Decafi','Italia',1,'12-12-1963')");
```

   - Execute the batch update

```
int [] updateCounts = stmt.executeBatch();
```

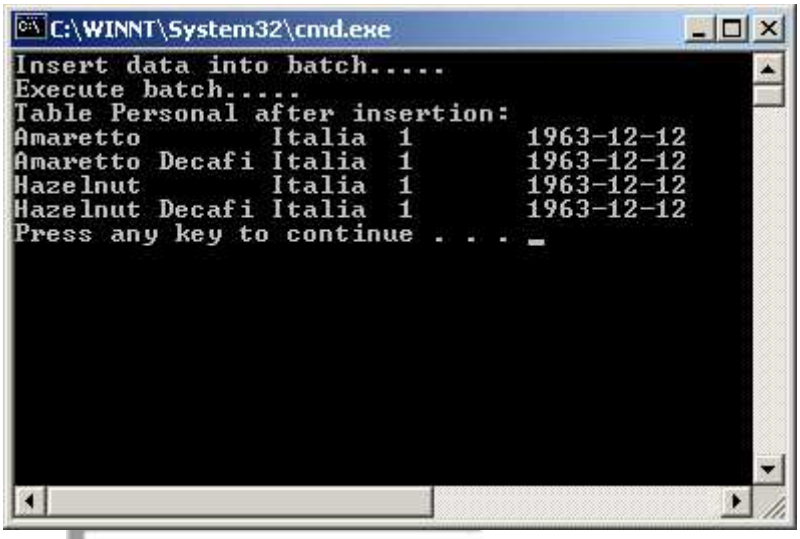


Figure 1

Do It Your Self :

2. Create program with this user interface bellow

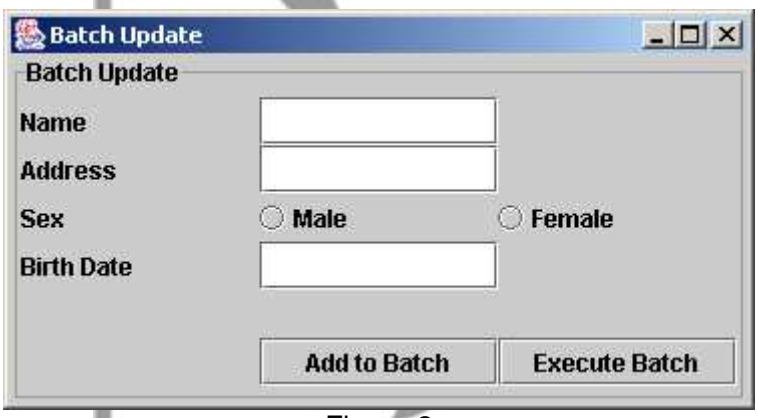


Figure 2

3.

- Fill the data in text file that are needed
- Press button **Add to Batch** to finish. This will add insert commanf into batch update.
- Press button **Execute Batch** to save all data into table Personal in database.

SESSION VI
Network Programming

Subject :
Socket and ServerSocket

Objective :
Understanding URL & URLConnection
Understanding Socket & ServerSocket

Review :
To create a URL with specific string that descript site location of a resource you can used :

```
URL(String spec)
```

while to construct a URL with specific protocol, host name, and port number you can used :

```
URL(String protocol, String host, int port, String file)
```

To open a connection to the URL object and returns an InputStream for reading from that connection, used this method

```
public final InputStream openStream( )
```

To create a stream socket and connects it to the specified port number at the specified IP address, use this constructor :

```
Socket(InetAddress address, int port)
```

To creates a server socket on a specified port, use this :

```
ServerSocket(int port)
```

Exercises :
Create two program (client & server program) that communicate trough network as follow :
Client will send request to server for searching data when user click Button Search. In respond to the client request server program will searching the data and will send the result back to the client. After receiving the result client will display the result to the textarea.

Guide for creating client program :
1. Using Swing create GUI as Follow :

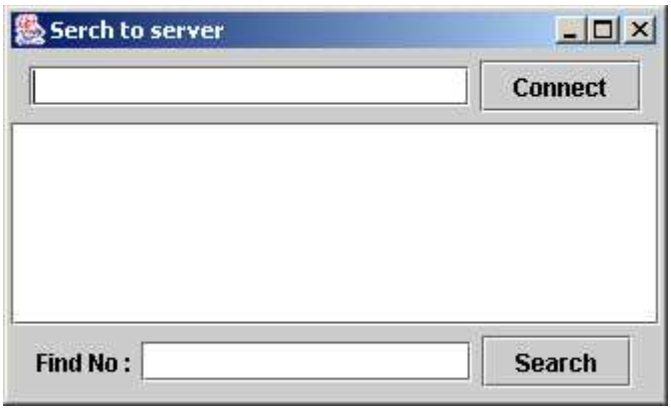


Figure 5.1
Client GUI

2. When user click Connect Button the client program will invoke method that build a communication Socket to server program with InetAddress 127.0.0.1 and port number 2003 as follow :

```
soc = new Socket("127.0.0.1", 2003);
tfStatus.setText(" connected ..");
oute = new PrintWriter(soc.getOutputStream(),true);
in = new BufferedReader(new
InputStreamReader(soc.getInputStream()));
```

please handle the error that may be thrown by those statement.

3. When the user clicked Button Send after the communication stream has been built the program will call a method that do as follow :
Send the String from the bottom side textfield to the server as a request for searching data. Wait for receiving search result from the server and display the result to the textarea. As follow :

```
      data = tfText.getText();
      tfStatus.setText("Searching ...");

      oute.println(data);

      String balik = in.readLine();
      System.out.println("Initial Reading OK "+balik);
      if(balik.equals("false")){
            tfStatus.setText("Sorry can't find those data !");
      }else{
            String datNIM = in.readLine();
            String datNama = in.readLine();
            taResult.setText("Search Result :\n");
            taResult.append("Enroll No    :
      "+datNIM+"\n");
            taResult.append("Name   :     "+datNama+"\n");
      }
```

please catch any error throws by any statement.

Guide for creating server program :

1. In main method of server program at first create ServerSocket on port number 2003, and then continousely create socket to handle client connection that accepted trough ServerSocket.

```
ServerSocket socketServer = null;

socketServer = new ServerSocket(2003);

System.out.println("listening ... ");
while(true){

   Socket socketClient = null;
   socketClient = socketServer.accept();
   System.out.println("accept conection ");
   new threadServis(socketClient).start();
```

You must handle any error thrown by any statement.

2. Notice on the last statement in the main method code above :

```
new threadServis( ).start( );
```

it means that the program call a class name threadServis, that is a Thread.

where the threadServis should implement all process performed by server program as responses of client request to the server (to search a Name of specified enroll number).
suppose that the data is stored in an Array. so the threadService class code is :

```
class threadServis extends Thread{
Socket socketClient;
String[] enroll = {"0001","0002","0003","0004","0005"};
String[] name = {"Anny","John","July","Rita","Andrew"};

public threadServis(Socket so){
      socketClient = so;
}

public void run(){
      BufferedReader in = null;
      PrintWriter oute = null;

      in = new BufferedReader(new InputStreamReader(
      socketClient.getInputStream()));
      oute = new PrintWriter(socketClient.getOutputStream(),
true);
      System.out.println("i/o OK server running ");

      String inputLine = null;
      inputLine = in.readLine();

      boolean hasil=true;
      int id=0;
      for(int i=0; i<5; i++){
            if(inputLine.equals(enroll[i])){
                  hasil = true; id = i;break;
            }else{ hasil = false;}
      }
      if(hasil){
            oute.println("true");
            oute.println(enroll[id]);
            oute.println(name[id]);
            System.out.println("search = "+hasil);}
      else{
      oute.println("false");System.out.println("saerch =
"+hasil);
      }
}
}
```

Do It Your Self :

Modify the Exercise Program above where the GUI as shown in figure 5.2. and the process that client program ask to the server program is as follow :
Client program will send two number entered by user in the textfield, to the server program will multiply those numbers and send back the result to the client. After receiving those result Client program will display in the textarea as shown in figure 5.2

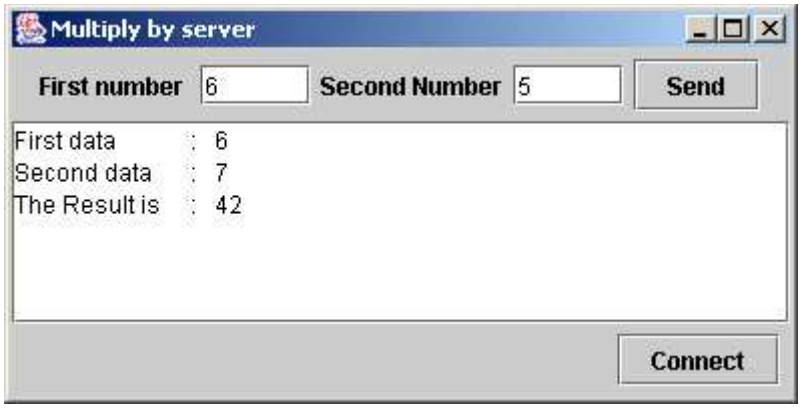


figure 5.2
Client GUI

SESSION VII
Network Programming

Subject :
DatagramSocket, DatagramPacket and MulticastSocket

Objective :
Understanding DatagramSocket
Understanding DatagramPacket
Understanding MulticastSocket

Review :
You can constructs a datagram socket and binds it to any available port on the local host machine, using this constructor :
DatagramSocket()
or you can constructs a datagram socket and binds it to the specified port on the local host machine.
DatagramSocket(int port)

To constructs a DatagramPacket for receiving packets with specific length. we can used :
DatagramPacket(byte[] buf, int length)

while to constructs a datagram packet for sending packets of specific length to the specified port number on the specified host. we used :
DatagramPacket(byte[] buf, int length, InetAddress address, int port)

To send a packet trough specified socket we can used method :
send(Datagrampacket pack);
of DatagramSocket class.
To receive packet from specified sacket we cant used method :
receive(DatagramPacket pack);
of DatagramSocket class.

Constructor : **MulticastSocket**(int port) is used to create a multicast socket and bind it to a specific port.
and to joint to a multicast group we cant used method : **joinGroup**(InetAddress addr);

Exercises :
Create two program (client program & server program) that communicate via network as follow :
when user run the client program, after create Client GUI the program will construct a DatagramSocket to communicate with server program and get InetAddress of specified Host that running server program. After user clicked the Button “Send” Client program will send the string to the server program trough the DatagramSocket. After receiving the packet the running server program will send back to the client program. Client program will display each receive packet from the server program.

Guide for creating Client program :

1. Using Swing create GUI as shown in figure 6.1.
2. After constructing all GUI component in constructor create DatagramSocket and get InetAddress using the follwing code :

```
comSocket = new DatagramSocket();
addr = InetAddress.getByName("comp1");
```

don't forget to catch any error thrown by the statement.

3. To implement processes when user clicked "Send" Button create a method that contain the following code :

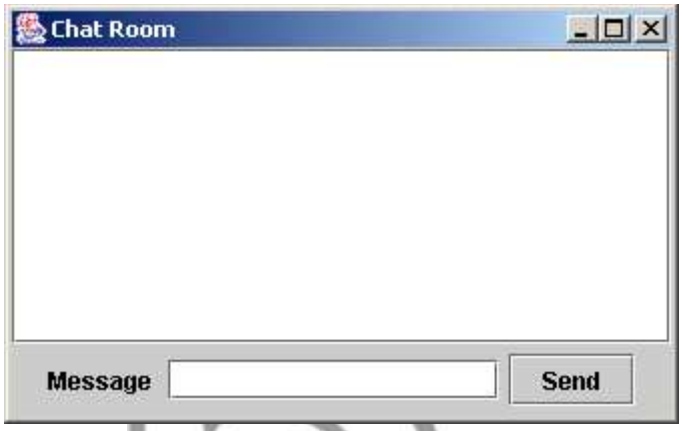


figure 6.1
Client GUI

```
DatagramPacket mpacket, recpacket;

byte[] buf2 = new byte[256];
buf2 = message.getBytes();
mpacket = new DatagramPacket(buf2, buf2.length, addr,
4500);
comSocket.send(mpacket);

recpacket = new DatagramPacket(buf2, buf2.length);
comSocket.receive(recpacket);

String data = new String(recpacket.getData());
taView.append(data);
```

don' t forget to catch any Exception thrown by each statement.

Guide for creating server program :

1. In main method of server program at first it will instant an object of serverGram class and call method start. serverGram is a class that extend a Thread class. we create serverGram class to implement all proses did by server.
2. In the constructor of serverGram class it will create a DatagramSocket object with port number 4500.

```
super("serverG");
socket = new DatagramSocket(4500);
```

serverGram class must override **run** method with the following code :

```
in = new BufferedReader(new
InputStreamReader(System.in));
try{
   String jalan = in.readLine();
   byte[] buf = new byte[256];
   System.out.println("waiting request ..");
```

// this loop is to run continuously all process of receiving a
// packet and sending back to request client

```
      while(!jalan.equals("exit")){
```

// this code is to receive request

```
DatagramPacket packet = new DatagramPacket(buf,
buf.length);
socket.receive(packet);
System.out.println("recieving request ..");
```

// this code is to send packet

```
String datasend = new String(packet.getData());
buf = datasend.getBytes();
InetAddress address = packet.getAddress();
int port = packet.getPort();
packet = new DatagramPacket(buf, buf.length, address,
port);
     socket.send(packet);
     System.out.println("sending packet ..");
}
in.close();
System.exit(0);
```

Do It Your Self :

Modify the program in the exercise so after server program receive packet from Client program, it will broadcast the packet to all running client program(send to all client that joint to the server)
please use MulticastSocket & method jointGroup in the Client Program, and use defined InetAddress for broadcast group.

```
comSocket = new MulticastSocket(4500);
addr = InetAddress.getByName("230.0.0.1");
comSocket.joinGroup(addr);

DatagramPacket packetsend = new DatagramPacket(buf,
buf.length, address, 4500);
socket.send(packetsend);
```

Session VIII

Java Bean

Subject
Creating Bean Component

Objective
1. Understanding Java Bean
2. Understanding steps creating Java Bean
3. Know how to use Java Bean

Creating Bean guide
1. Write a java bean code with the following main criteria:
    a. If the class have constructors it must a default constructor
    b. Each properties should have set and get method
    c. The class should implements Serializable interface.
2. Compile the Bean class above
3. Create manifest file for the Bean class. The manifest file should contained :
   Java-Bean: True
   Name: BeanName.class
4. Create jar file. The jar file contains manifest file and Bean classes
5. The created jar file should located in the classpath. Its ready to use.

Exercises
1. Create bean class, the bean is a message box contains a message, title and Button. As shown bellow:

2. Create java program that use the bean. This program is a frame that have a button. If user click the button the program will invoke the bean and show the message box.

Do it your self
1. Create bean class, the bean is a Input box contains a message, title, text filed to enter user input and Button.

2. Create java program that use the bean. This program is a frame that have a button. If user click the button the program will invoke the bean and show the input box. After user click ok button the program should get the user input and view the input in the parent frame.

Session IX

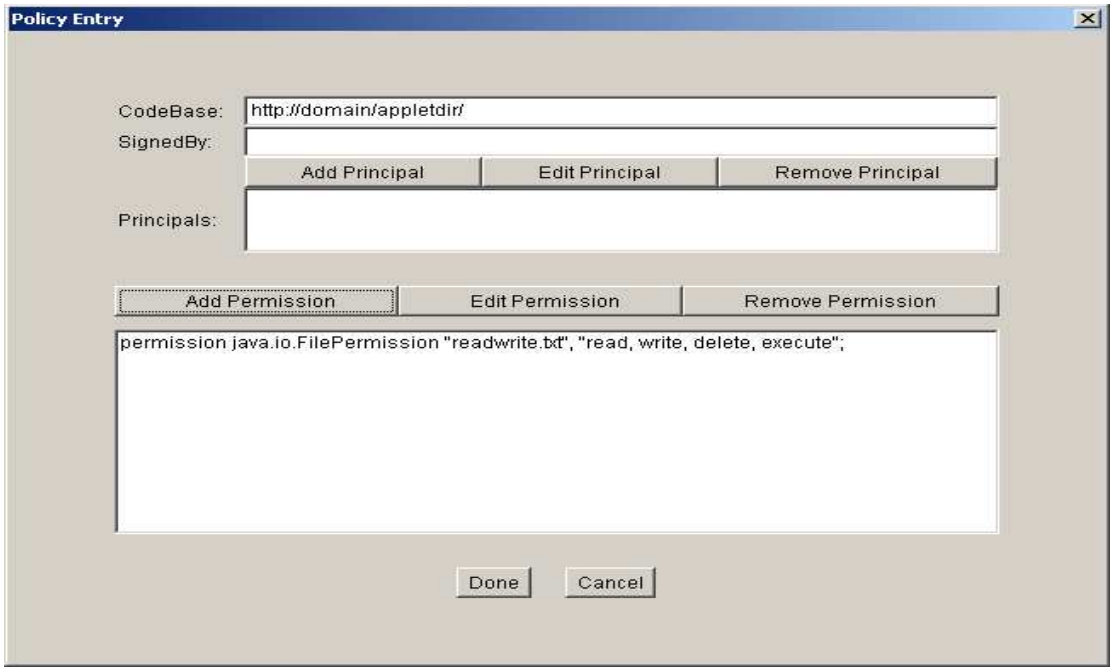
Java Security

Subject
Controlling Applet and Create Digital Signature

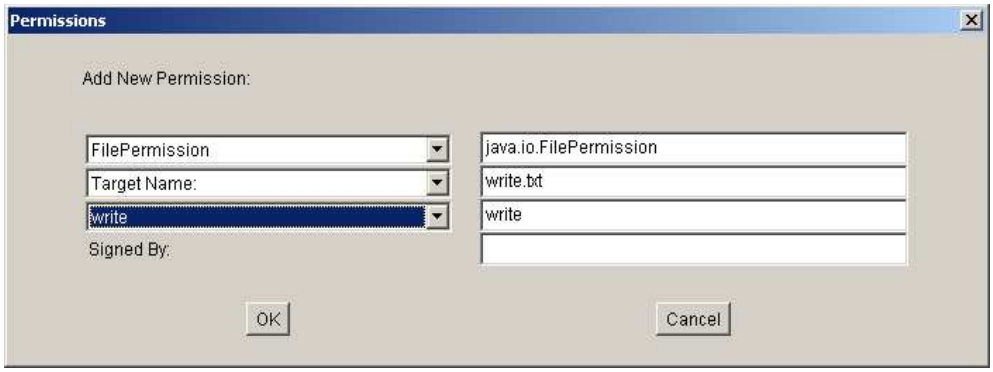
Objective
1. Familiar with java security policy
2. Able to create jar file with digital signature

Java security guide
Creating policy file
1. Run the **policytool** program

2. Add new policy with the code base **http://hostname/appletdirectory/** or **file:/c:/foldername/**



3. Add permission



4. Save the policy file
   Chose Menu file and Save with the name **policyfile**
5. To see the result, create an applet that accessing file in local computer, run appletviewer with the following command

Appletviewer –J-Djava.security.policy=policyfile
http://hostname/appletdirectory/appletname.html

Creating jar file with digital signature

a. Steps for code signer
   1. Creating jar file contain the class file
      jar cvf FileName.jar ApplicationName.class

   2. Generate the public and private key pair with the public key in certificate
      keytool -genkey -alias signFiles -keypass kyp789
      -keystore nimmistore -storepass ap987t

3. Sign the jar file
   jarsigner -keystore nimmistore -signedjar sFileName.jar FileName.jar signFiles

4. Export the public key certificate
   keytool -export -keystore nimmistore -alias signFiles -file NimmiRamirez.cer

b. Steps for code receiver
   1. Observe the restricted application
      java -Djava.security.manager -cp sCount.jar Count C:\TestData\data

   2. Import the certificate as trusted certificate
      keytool -import -alias nimmi -file NimmiRamirez.cer -keystore raystore

   3. You can get the finger print with the following command
      keytool -printcert -file SusanJones.cer

   4. Set up policy file to grant the required permission using policytool

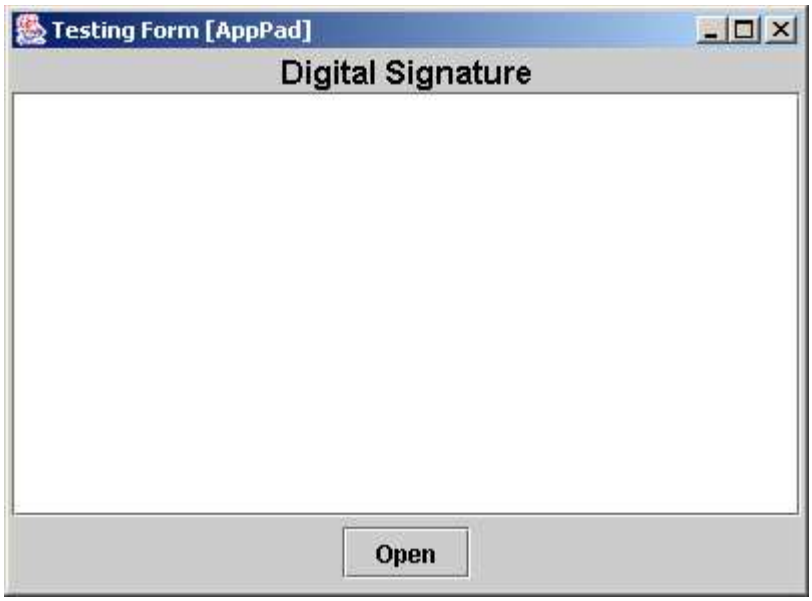
Exercises

1. Create file c:\temp**data**. This file must contain text as shown in the following figure.

2. Create an applet as shown in figure … above, when user click Open button in the applet then it's will display contents of file in the applet text area. And when user click Save button all text in the applet text area will store to the file
3. Create policy file to grant access to those file.

Do it your self

1. Create an application that read file from local disc **c:\temp\data** and showing in the text area



2. Create jar file from those class file **Application.jar**
3. Sign the jar file sApplication.jar and create certificate **java.cer**
4. Copy the signed jar file and certificate to **c:\digital**
5. Run the application under java security manager.

SESSION X
Remote Method Invocation : Basic Concepts

Objective :

- Understand remote method invocation mechanism
- Capable of creating a simple RMI Application

Exercise :

1. Create and compile a class representing a Person entity, with the following properties :
- full name
- address
- sex

and the respective accessor and mutator method, as follow :

```
public class Person implements Serializable
{
  private String fullName;
  private String address;
  private String sex;

  public String getName() {}
  public String getAddress() {}
  public String getSex() {}

  public void setName(String name) {}
  public void setAddress(String address) {}
  public void setSex(String sex) {}
}
```

2. Create and compile a remote interface with one method for saving Person object to permanent storage.
The code is provided as follow :

```
public interface PersonData extends Remote
{
  public boolean save(Person p) throws RemoteException ;
}
```

3. Create a remote class, implementing method save in the interface. The method will write data in Person object to database.

```
public class PersonDataImpl extends UnicastRemoteObject
                            implements Remote
{
  public PersonDataImpl()
  {
    super();
    // initialize database connectivity here ...
  }

  public save (Person p)
  {
    // insert person data into database here ...
  }
}
```

4. Create a client class accessing remote method save, passing your personal data encapsulated in Person object.

```
public class ClientApps
{
  public static void main(String[] args)
  {
    // install security manager
    // lookup remote object
    // invoke save
  }
}
```

5. Create a class for Remote Object registration and activation.

```
public class ServerRunner
{
  public static void main(String[] args)
  {
    // instantiate remote implementation object
    // start rmiregistry
    // register remote object to rmiregistry
  }
}
```

6. Compile the java files

```
$ javac *.java
$ rmic PersonDataImpl
```

7. Place the class files in the appropriate place

8. Compile and Run the application

```
$ java ServerRunner
```

Do it yourself :

- Extend the application by adding method for deleting a person data from the database.

SESSION XI
Remote Method Invocation : Dynamic Class Loading

Objective :

- understand concept and architecture of dynamic class loading mechanism
- able to implement dynamic class loading feature in Java

Exercise :

1. Create a shared folder accessible from network. Test the shared folder for downloading file. The shared folder will be referred as **`$shared`** in this example. Replace **`$shared`** with path to your shared folder.

2. The server invoker, remote interface and implementation class remains unchanged. Compile and place the class file and the stub in the shared folder.

3. The client will be loaded dynamically. Create class for invoking client application. The code is provided as follow:

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.Properties;

public class ClientLoader{
  public static void main(String[] args){

    // installing security manager
    if (System.getSecurityManager() == null) {
      System.setSecurityManager(new RMISecurityManager());
    }

    try {

      // getting properties setting
      Properties p = System.getProperties();
      String clientClassUrl =
              p.getProperty("java.rmi.server.codebase");

      // loading client class
      Class clientClass = RMIClassLoader.loadClass(
                              clientClassUrl, "HelloClient");
      clientClass.newInstance();
    } catch (Exception err) {
      err.printStackTrace();
    }

  }
}
```

4. Compile and place the class file in your local folder. We will invoke this class locally.

5. The client application itself must be configured to be able to load dynamically, create a constructor in client apps as follow:

```
import java.rmi.*;
import java.rmi.server.*;

public class DynamicHelloClient{
  DynamicHelloClient(){
    System.setSecurityManager(new RMISecurityManager());
    String url = "rmi://comp11:2003/hello";

    try {
      System.out.println("Searching hello .. ");
      Hello h = (Hello) Naming.lookup(url);
      h.sayHello();
      System.out.println(
                 "Method invoked, check server output");
    } catch (Exception err){
      err.printStackTrace();
    }
  }
}
```

6. Compile and place the client apps' class file in **`$shared`**.

7. Run the server-invoker class (ServerRunner)

8. Run the client class with specified property setting, as follow:

```
$ java -Djava.security.policy=hello.policy
       -Djava.rmi.server.codebase=$shared
       ClientLoader
```

Do it yourself :

Extend the application so that the server class is dynamically loaded in the runtime as well.

SESSION XII
Remote Method Invocation : Activation

Objective :

- Understand object activation and passivation concepts
- Understand object activation and passivation mechanism

Exercise :

1. Create a remote interface with one method capable of calculating addition of two integer.

2. Create a client accessing and invoking the remote method

3. Create a remote class using Activatable as the superclass and implementing remote interface.

```
public class ActiveImpl extends Activatable implement
ActiveInterface
{
  public ActiveImpl (ActivationID id, MarshalledObject data)
                                      throws RemoteException
  {
    super (id, 0);
  }

  // implement remote method here
}
```

4. Create another class for loading the remote implementation.

```
public class ActiveRunner {
  public static void main (String[] args) {
    try {
      // install security manager here

      // Creating the group
      String myPolicy = "path to your policy file";
      Properties p = new Properties();
      p.put("java.security.policy", myPolicy);

      ActivationGroupDesc groupDesc
                  = new ActivationGroupDesc(p, null);
      ActivationGroupID groupID
                  = ActivationGroup.getSystem().
                                  registerGroup(groupDesc);
      // creating object description
      String classDir = "path to your classfile location";
      ActivationDesc objectDesc =
          new ActivationDesc("ActiveImpl", classDir, null);
      ActiveInterface remoteObj =
         (ActiveInterface)Activatable.register(objectDesc);
      // Registering object
      Registry reg = LocateRegistry.createRegistry(2003);
      reg.rebind("ActiveServer", remoteObj);
    } catch (Exception err) {
      err.printStackTrace();
    }
  }
}
```

5. Compile all classes, put remote interface, remote implementation, and stub classes in shared folder.

6. Run the application

Do it yourself :

Modify your Person management application to comply with the exercises above.

Sesssion XIII

Servlet Programming

Subject
Basic Servlet Programming and Session Handling

Objective
- Web Container configuration
- Undestanding servlet life Cycle
- Familliar with handling request and response using doGet() and doPost() method
- Database connectivity with servlet
- Familiar with session handling

Servlet Configuration Guide
Web aplication structure:
- Public Directory
- WEB-INF/web.xml file
- WEB-INF/classes directory
- WEB-INF/lib directory

Creating servlet program using Tomcat web container step by step
1. Locating web application foolder
   Example:
```
C:\ServletProgram
                    \src
                        \ServletTest.java
                    \index.html
                    \WEB-INF
                            \web.xml
                            \classes
                                    \ServletTest.class
```
2. Edit Tomcat configuration file %TOMCAT_HOME%\conf\server.xml and add new context as follow:
```
<Context path="ServletProgram"
     docBase="C:\ServletProgram">
</Context>
```
3. Create html file C:\ServletProgram\index.html
4. Create servlet code program
   C:\ServletProgram\src\ServletTest.java

5. Compaile the Servlet code program and copy to \WEB-INF\classes directory.
   C:\ServletProgram\WEB-INF\classes\ServletTest.class

6. Create Deployment Descriptor web.xml file bellow:
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<DOCTYPE web-app
 PUBLIC "-//Sun Microsystem, Inc.//DTD Web Application
   2.2//EN"
 "http://java.sun.com/j2ee/dtds/web-app_2.2.dtd">
<web-app>
     <servlet>
```

```
            <servlet-name>Test</servlet-name>
            <servlet-class>ServletTest</servlet-class>
      </servlet>
      <servlet>
            <servlet-name>Test</servlet-name>
            <url-pattern>/servlet/Test</url-pattern>
      </servlet>
</web-app>
```

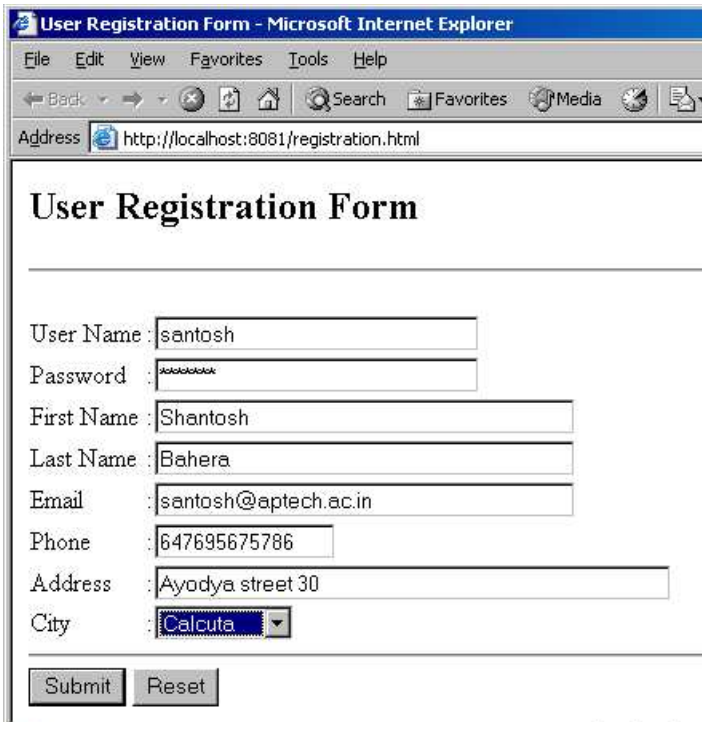
7. You can open web browser and type the following address to see the result http://hostname:8080/ServletProgram/Test

Exercises

1. Create database: **ServletDb**
2. Create **members** table

Field	Type	Key
User_name	Varchar(40)	Primary key
Password	Varchar(40)	
First_name	Varchar(30)	
Last_name	Varchar(30)	
Email	Varchar(40)	
Phone	Varchar(15)	
Address	Varchar(50)	
City	Varchar(40)	

3. Create file **registration.html** as follow



4. Create **Register** servlet. This servlet is to process of restoring data to database as a response of your request. The servlet will displaying page as shown the following figure when the restoring process finished.

Do It Your Self

1. Create file login.html as follow:

2. Create **CheckLogin** servlet to validate user name and password, if login successfull then servlet will display hyperlinks to user profile servlet and logout servlet.

if login failed the servlet will display the following message "**Invalid username or password!**"

3. Create the **Profile** servlet to display active user details at this time. And this servlet can't be accessed if the user not login.

4. Create **Logout** servlet that handling logout session (invalidate the session).

Session XIV

Servlet Programming

Subject
Context dan Collaboration

Objective
- Understanding servlet Context
- Understanding request dispatcher

Servlet Context Configuration Guide
To add context parameter you can edit the deployment descriptor file (web.xml file) as follow:

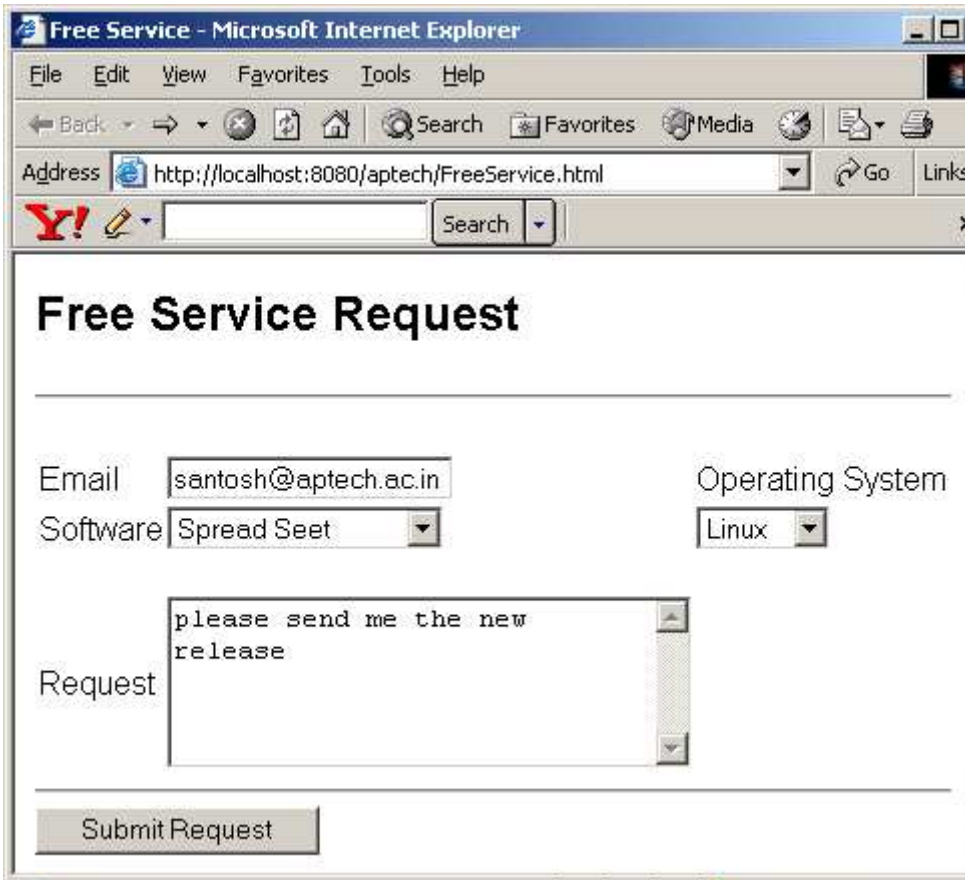
```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <context-param>
     <param-name>driver</param-name>
     <param-value>
     com.microsoft.jdbc.sqlserver.SQLServerDriver
     </param-value>
  </context-param>
  <context-param>
    <param-name>protocol</param-name>
    <param-value>
     jdbc:microsoft:sqlserver://win2ksvr:1433;
     DatabaseName=servletDb;User=sa;Password=
    </param-value>
  </context-param>
  <servlet>
    <servlet-name>Banner</servlet-name>
    <servlet-class>Banner</servlet-class>
  </servlet>
     .
     .
     .
<web-app>
```

Exercises
1. Create **Service** table with the fillowing structure

Field	Type	Key
Id	Int	Primary key(Auto Increament)
email	Varchar(40)	
softwar e	Varchar(40)	
os	Varchar(30)	
request	Varchar(50)	

2. Create file freeservice.html as follow:



3. Create RequestServlet servlet to handle request and from freeservice.html page and restore data to service table. Use getServletContext() method to get context parameter (driver and url) containing in the web.xml file
4.

Do It Your self

1. modify RequestServlet file to check weather email of requested user is already stored in the members table. If user email is exist in the members table the servlet will forward request to ResponseServlet with request attribute lastName and firstName retrieving from members table.

2. Otherwise it will forward to register.html with session attribute email address.