Laboratory Manual

For

Advanced Java Technology

(IT)

B.Tech (IT)

SEM VI

July 2016

Faculty of Technology
Dharmsinh Desai University
Nadiad.
www.ddu.ac.in

Table of Contents

Sample Experiment

1 AIM:

Create a GUI based application which can be used as a telephone directory application. The telephone directory is stored as a database and has one table named telephoneDir. The telephoneDir database table stores three different information: telephone no., owner name, and owner address. The owner name is made of three parts: First name, middle name, and last name. The owner address is made of five parts: house no., address 1, address 2, area name, and city name. The application allows search facility. The search is possible using three different ways:

1. Search by telephone no.
2. Search by name (one of first name, middle name, and last name) with exactly match and part of name.
3. Search by address (one of address 1, address 2, area name, and city) with exactly match and part of address.

2 TOOLS/APPARATUS:

NetBeans IDE 6.7

3 STANDARD PROCEDURES:

a. Analyzing the Problem:

Identify required classes, methods and database and write the logic.

b. Designing the Solution:

Step:1

Create a table with appropriate column names and proper GUI as per the user's requirement.

Step:2

Implement the code to connect with database.

Step:3

Implement the code as per the given application.

Step:4

Check the output by providing various inputs and test the output as per the application.

Class Diagram:

Telephone Directory
Telephone Number
Owner_FirstName
Owner_Middle Name
Owner_Last Name
Owner_ Address1
Owner_ Address2
Owner_ HouseNo
Owner_ AreaName
Owner_ City
Owner_Contact

ActionPerformed(); ItemStateChange();

GUI:

Searching options:

Search By Telephone No
Search By Name
Search By Address

Submit

Telephone No: 1234567890

o/p:

3.3 Implementing the Solution

3.3.1 Writing Source Code

```
package lab1;

import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class TelephoneDirectory extends Frame implements ItemListener,ActionListener
{

   Panel pTop=new Panel();
   TextField tf1=new TextField(20);
   TextArea ta1=new TextArea(10, 100);
   Choice c1=new Choice();
   Choice c2=new Choice();
   Button srchbtn=new Button("Search");
   Label status=new Label("Records Found = 0");
   String query=null;

   Connection cn=null;
   PreparedStatement pstat=null;
   ResultSet rs=null;

   public TelephoneDirectory()
   {
      super("My Telephone Directory");
      setVisible(true);
      setSize(800,400);

      c1.add("Telephone No");
      c1.add("Name");
      c1.add("Address");
      c1.addItemListener(this);
      srchbtn.addActionListener(this);

      addWindowListener(new WindowAdapter(){
         public void windowClosing(WindowEvent e)
         {
            dispose();
         }
      });

      setLayout(new BorderLayout());
```

```
        pTop.setLayout(new GridLayout(4, 2));
        pTop.add(new Label("Search Option 1 : "));
        pTop.add(c1);
        pTop.add(new Label("Search Option 2 : "));
        pTop.add(c2);
        c2.setVisible(false);
        pTop.add(new Label("Enter Text : "));
        pTop.add(tf1);
        pTop.add(new Label(""));
        pTop.add(srchbtn);
        add("North",pTop);
        add("Center",ta1);
        add("South",status);
        ta1.setEditable(false);

        // Initialization of Connection Object

        String url="jdbc:odbc:teledsn";
        String driver="sun.jdbc.odbc.JdbcOdbcDriver";

        try{
           Class.forName(driver);
           cn=DriverManager.getConnection(url);
        }
        catch(ClassNotFoundException e)
        {
           System.out.println(""+e.toString());
        }
        catch(SQLException se)
        {
           while(se!=null)
           {
              System.out.println(""+se.toString());
              se=se.getNextException();
           }
        }
    }

    public void itemStateChanged(ItemEvent ee)
    {
        String arg=ee.getItem().toString();
        if(arg.equals("Telephone No"))
        {
           c2.setVisible(false);
```

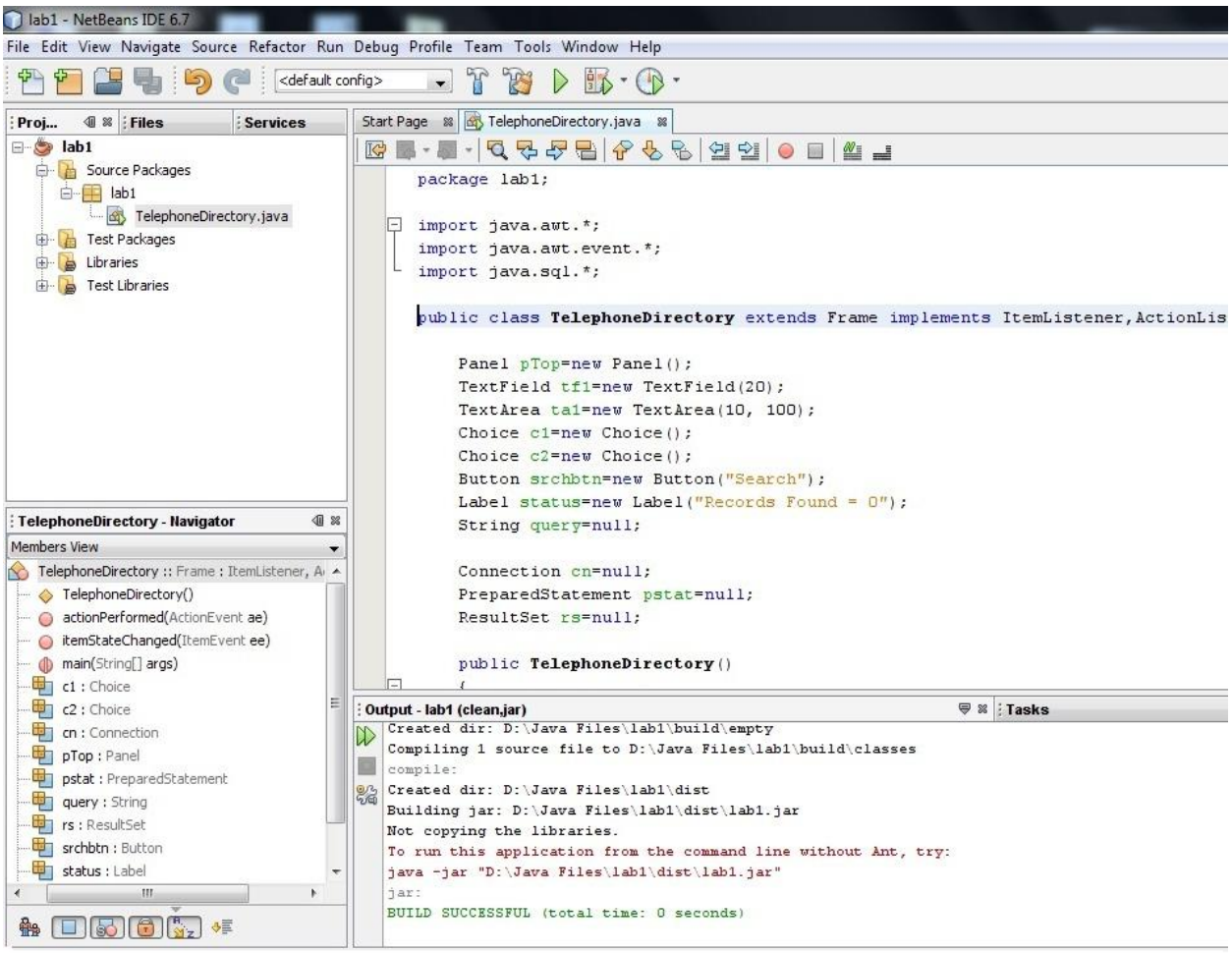
```
        }
        else if(arg.equals("Name"))
        {
           c2.removeAll();
           c2.add("First Name");
           c2.add("Middle Name");
           c2.add("Last Name");
           c2.setVisible(true);
        }
        else if(arg.equals("Address"))
        {
           c2.removeAll();
           c2.add("Area");
           c2.add("City");
           c2.setVisible(true);
        }
     }

     public void actionPerformed(ActionEvent ae)
     {
        ta1.setText("Refreshed");
        query=new String("select * from Directory");
        int len=0;
        len=tf1.getText().toString().trim().length();

        try
        {
           if(c1.getSelectedItem().equals(("Telephone No")) && len>0)
           {

              query+=" where Number=?";
              pstat=cn.prepareStatement(query);
              pstat.setString(1, tf1.getText().toString().trim());
           }
           else if(c1.getSelectedItem().equals(("Name")) && len>0)
           {
              if(c2.getSelectedItem().equals(("First Name")))
                 query += " where First_Name=?";
              else if(c2.getSelectedItem().equals(("Middle Name")))
                 query += " where Middle_Name=?";
              else if(c2.getSelectedItem().equals(("Last Name")))
                 query += " where Last_Name=?";

              pstat=cn.prepareStatement(query);
```

```
            pstat.setString(1, tf1.getText().toString().trim());
        }
        else if(c1.getSelectedItem().equals(("Address")) && len>0)
        {
            if(c2.getSelectedItem().equals(("Area")))
                query += " where Area=?";
            else if(c2.getSelectedItem().equals(("City")))
                query += " where City=?";

            pstat=cn.prepareStatement(query);
            pstat.setString(1, tf1.getText().toString().trim());
        }
        else
        {
            pstat=cn.prepareStatement(query);
        }

        try
        {
            System.out.println(query);
            rs=pstat.executeQuery();
        }catch(NullPointerException ne)
        {
            System.out.println("Text Null3");
            ta1.setText("No Records Found.");
            status.setText("Records Found = 0");
        }
        if(rs!=null)

    ta1.setText("Number\t\tFName\t\tMName\t\tLNAME\t\tAdd1\t\tAdd2\t\tArea\t\tCity\n");

        int count=0;
        while(rs.next())
        {
            ta1.append("" + rs.getString(1) + "\t");
            ta1.append("" + rs.getString(2) + "\t");
            ta1.append("" + rs.getString(3) + "\t");
            ta1.append("" + rs.getString(4) + "\t");
            ta1.append("" + rs.getString(5) + "\t");
            ta1.append("" + rs.getString(6) + "\t");
            ta1.append("" + rs.getString(7) + "\t");
            ta1.append("" + rs.getString(8) + "\t\n");
            count++;
        }
```

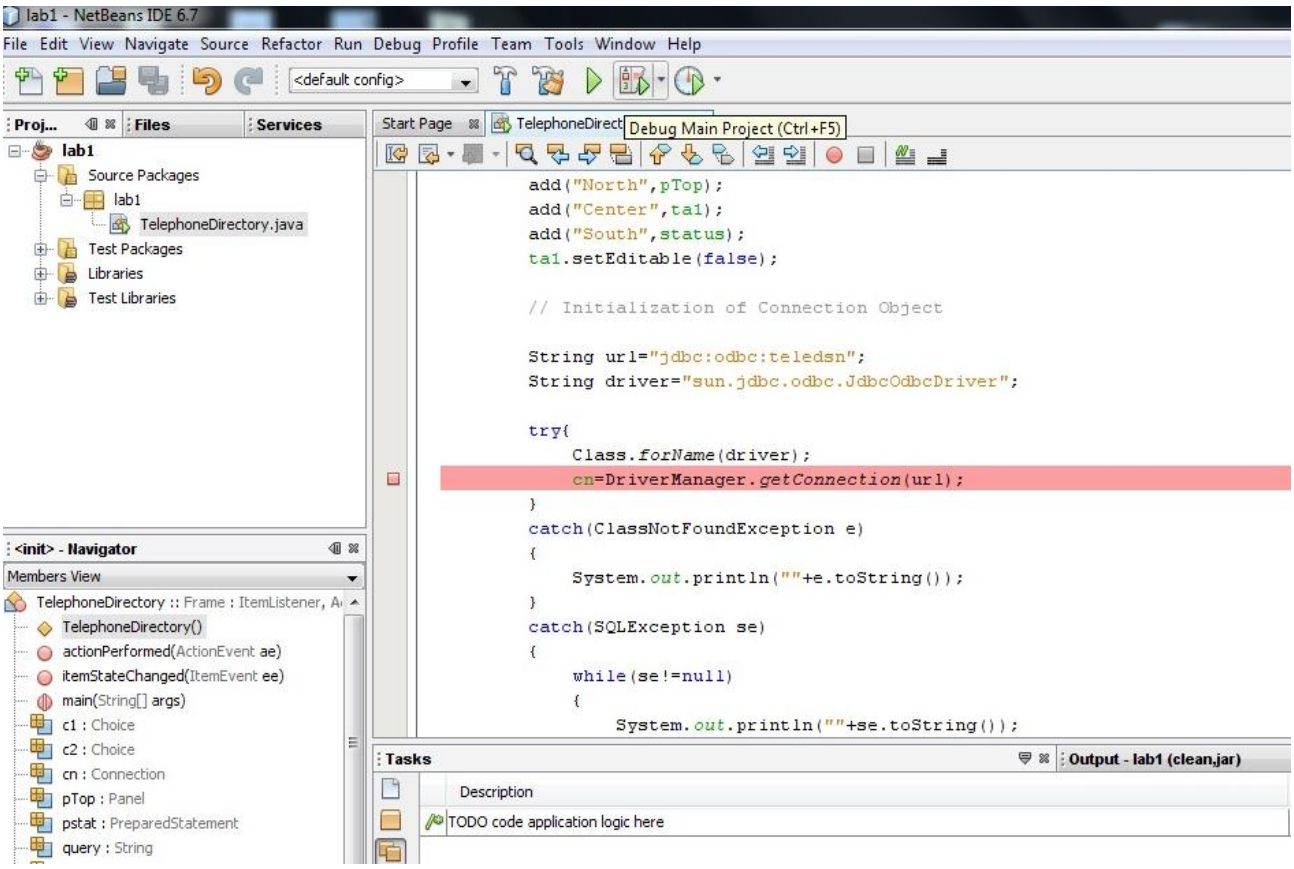
```
            status.setText("Records Found = " + count);
        }
        catch(Exception ee){
            System.out.println("Exception " + ee);
        }
    }
    public static void main(String[] args) {
        // TODO code application logic here
        Frame dir=new TelephoneDirectory();
    }
}
```

3.3.2 Compilation /Running and Debugging the Solution

To compile the application: Right Click on your project in project-explorer and click on build. Or click on bulid project submenu from run menu. To run the application click on Run project submenu from run menu or press f6.



Debug: To debug the application, keep the break points and click on debug menu.


lab1 - NetBeans IDE 6.7
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Proj... Files Services
lab1
Source Packages
lab1
TelephoneDirectory.java
Test Packages
Libraries
Test Libraries
Start Page
TelephoneDirect
Debug Main Project (Ctrl+F5)
add("North",pTop);
add("Center",ta1);
add("South",status);
ta1.setEditable(false);
// Initialization of Connection Object
String url="jdbc:odbc:teledsn";
String driver="sun.jdbc.odbc.JdbcOdbcDriver";
try{
Class.forName(driver);
cn=DriverManager.getConnection(url);
}
catch(ClassNotFoundException e)
{
System.out.println(""+e.toString());
}
catch(SQLException se)
{
while(se!=null)
{
System.out.println(""+se.toString());
<init> - Navigator
Members View
TelephoneDirectory :: Frame : ItemListener, A
TelephoneDirectory()
actionPerformed(ActionEvent ae)
itemStateChanged(ItemEvent ee)
main(String[] args)
c1 : Choice
c2 : Choice
cn : Connection
pTop : Panel
pstat : PreparedStatement
query : String
Tasks
Output - lab1 (clean,jar)
Description
TODO code application logic here

c. Testing the Solution

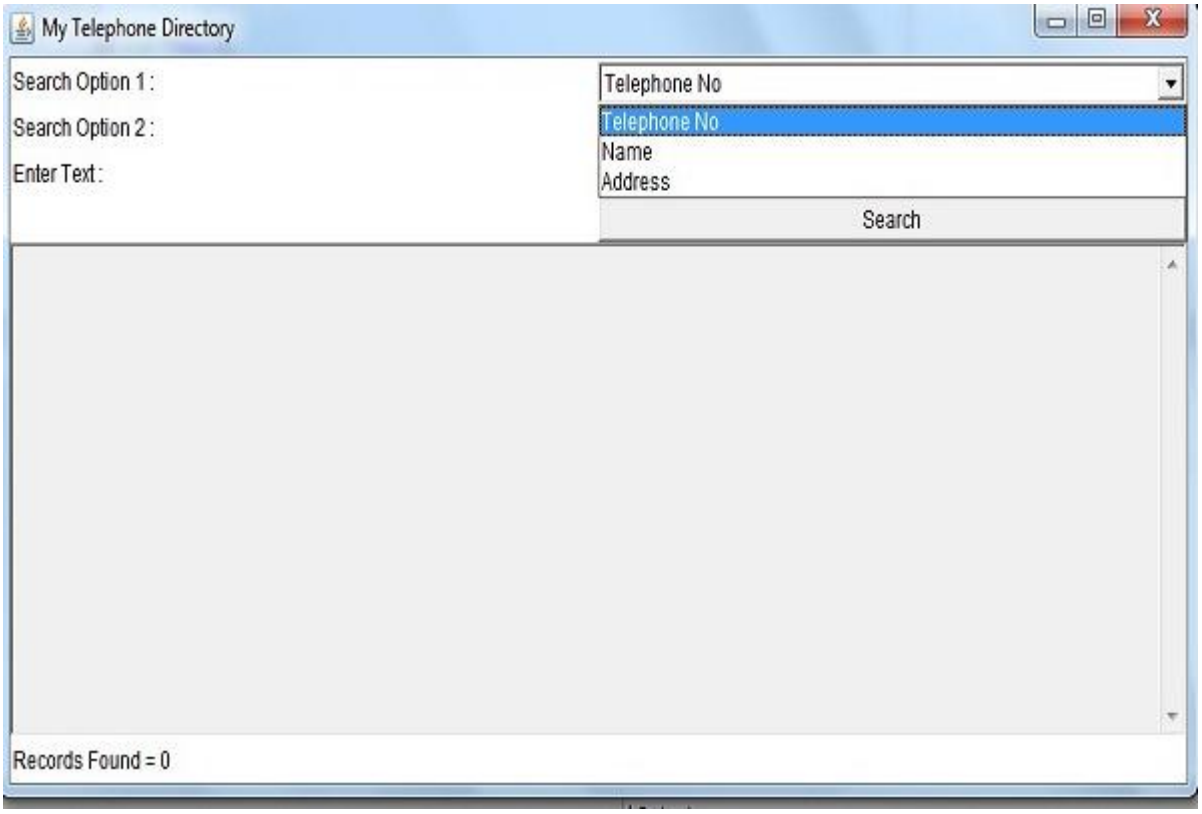
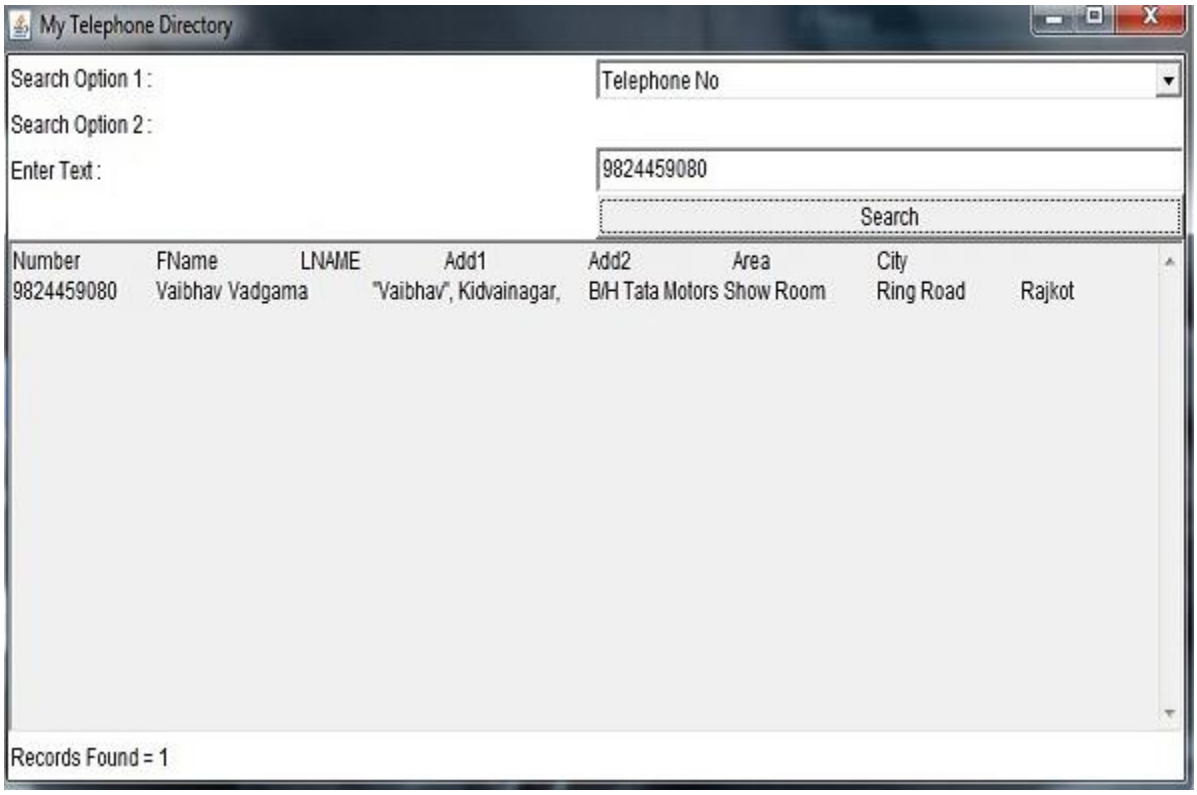


Figure 1: **Record Searching by (Telephone Number)**



Here, 1 record is found as per the given search data. Same way, searching can be done by owner name and owner address. User will enter owner name or address and appropriate data record will be displayed to the user.

COMMON PROCEDURES AND TOOLS REQUIRED FOR ALL THE EXPERIMENTS

Required Software/ Software Tool

Java Development Kit (JDK)
Install available current version of JDK.
Editor (Notepad)
Netbeans IDE 6.1
Install available current version of above software tools.

COMMON PROCEDURE:

Step 1: For the given problem statement design
Flowchart/Algorithm/Logic

Step 2: Define class name(s), find out appropriate attributes and create the methods which will show the flow of the program.

Step 3: Write the program code in appropriate different files having suitable extension. While saving code in a file consider rules for giving name to file.

Step 4: Compile java source code.

Step 5: Deploy, run and if any errors debug your program by debugger which is available in your editor.

Step 6: Test your program using sample input and write down output.

EXPERIMENT – 1

Topic: JDBC

Aim: Create a GUI based application which can be used as a telephone directory application. The telephone directory is stored as a database and has one table named telephoneDir. The telephoneDir database table stores three different information: telephone no., owner name, and owner address. The owner name is made of three parts: First name, middle name, and last name. The owner address is made of five parts: house no., address 1, address 2, area name, and city name. The application allows search facility. The search is possible using three different ways:

4. Search by telephone no.
5. Search by name (one of first name, middle name, and last name) with exactly match and part of name.
6. Search by address (one of address 1, address 2, area name, and city) with exactly match and part of address.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1

Procedure:

- Create Database
- Create a new DSN
- Create GUI
- In button click event :-
    - Load the JDBC driver using Class.forName() method.
    - Get the connection object using DriverManager.getConnection() method
    - Open the connection to the database.
    - Create a statement object using createStatement() method of connection object.
    - Write a select query and pass it as an argument to executeQuery() method of statement object.
    - Process the results returned in the resultSet object.
    - Close the connection to the database.
- Display the returned details to the user in a textarea.

EXPERIMENT – 2

Topic: JDBC

Aim: Create a GUI based application which can be used for telephone directory modification (administrator part for the above problem statement). The application allows two modification operations: create new telephone connection, and delete a telephone connection. The insert operation takes telephone no., name, and address as input parameters. The delete operation has verification step in which the user must perform the verification of the telephone connection which is about to be deleted. Once the verification is done, the application allows deleting the telephone connection. Design appropriate GUI to accommodate all stated features.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1

Procedure:

- Use Database of Experiment-1
- Use DSN of Experiment-1
- Create GUI
- In button click event :-
    - Load the JDBC driver using Class.forName() method.
    - Get the connection object using DriverManager.getConnection() method
    - Open the connection to the database.
    - Create a statement object using createStatement() method of connection object.
    - Write the prorer insert/delete query and pass it as an argument to executeUpdate() method of statement object.
    - In case of delete operation create a dialog box that asks for user confirmation and then only the delete operation must be performed.
    - Close the connection to the database.
- Display the status of insert/delete to the user.

EXPERIMENT – 3

Topic: RMI

Aim: Create a RMI based client-server application. The server allows access of bank account object to client through RMI mechanism. The account object allows following operations: deposit, withdraw, and balance. The server stores account data in database. Design appropriate interface and test implementation on network.

Tools /Apparatus: Editor (Notepad), command prompt.

Procedure:

- Create a server side interface java file which will expose the following methods: Deposit(), Withdraw(), showBalance().
- Create server side interface implementation java file for the interface created in the above step by implementing the above mentioned methods.
    - Deposit( ) :- JDBC process for update query
    - Withdraw( ) :- JDBC process for update query
    - showBalance( ) :- JDBC process for select query
    - Create a implementation object and register it to the RmiRegistry using Naming.Rebind() method.
- Create a main server java file
- Create a main client java file
    - Lookup the remote object in the RmiRegistry using Naming.lookup() method and using that reference call the remote method on the remote object.
- Compile all the java classes using **javac**
- Create stub and skeleton using **rmic**
- Start the registry by using **start rmiregistry**
- Open another command prompt and start the server using **start serverclassfile**
- Open another command prompt and start the client using **java clientclassfile**

EXPERIMENT – 4

Topic: Servlets

Aim: Create user registration functionality for student to get registered with exam- result section. The registration page takes following information from user: user ID, password, confirm password, full name, semester, roll no, email-id, and contact number. The registration servlet checks uniqueness of user ID among all users and if found unique then only stores registration information in database.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1, Web Browser

Procedure:

- Create a index.html file in which write html code for displaying your webpage with appropriate webcontrols.
- Create a servlet class file
    - Inside the doPost() method :-
    - Copy the required parameters into local variables using getParameter() method of request object
    - Perform JDBC process for select query by specifying the passed userID in the where clause
    - Create a printwriter object using the getWriter() method of response object.
- If any records found for the given userID then return an appropriate error message else return a registration confirmation message to the end user by writing the html code in the servlet class file using the printwriter object.
- Modify web.xml file according to the needs of the application.

EXPERIMENT – 5

Topic: Servlets: session management

Aim: Create login and view result functionality with the session management. The login servlet logons the user with the exam-result section and allows access of viewing his/her exam-result

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1, Web Browser

Procedure:

- Create a index.html file in which write html code for displaying your webpage with appropriate webcontrols.
- Create a servlet class file
    - Inside the doPost() method :-
    - Copy the required parameters into local variables using getParameter() method of request object
    - Perform JDBC process for select query by specifying the passed userID in the where clause to check the existence of the userID.
    - Get the HttpSession object.To obtain a session, use the getSession() method of the ServletRequest object Assume the session management facility uses cookies.
    - Store and retrieve user-defined data in the session.
    - Output an HTML response page containing data from the HttpSession object using the printwriter object.
    - End the session.
- Show the detailed result to the user if he/she is registered or else return an error message.
- Modify web.xml file according to the needs of the application.

EXPERIMENT – 6

Topic: JavaBeans

Aim: Create a JavaBean to store information about person. The details of person (person name, person age, person height, etc.) are stored in person database table. After the person is authenticated, his/her personal details are transferred from the database table (person) to JavaBean (Person) and the details are displayed in proper format using this Person JavaBean. The JavaBean is stored in session scope.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1, Web Browser

Procedure:

- Create a index.jsp page to display appropriate GUI for user and use the tag for java bean in this page.
- Create another JSP page for processing the authenticity of the user by calling the relevant method of the bean class using the object returned by getProperty field of jsp tag.
- Create yet another JSP page for displaying the relevant details to the user by calling the relevant method of the bean class using the object returned by getProperty field of jsp tag.
- Create a bean class which will perform database connectivity as well as define the getter and setter methods in this bean class.

EXPERIMENT – 7

Topic: JavaServer Pages, JSTL (core, sql)

Aim: Create a JSP based Web application which allows the user to edit his registration information (Refer EXPERIMENT-4). If login is successful, the user authentication servlet creates the welcome message for the user in session scope and then forwards the request to JSP page which handles the edit operation. Use the JSTL core library for variable creations, use and iterations, and JSTL SQL library for interaction with the database.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1, Web Browser

Procedure:

- Create a index.jsp file in which write html code for displaying your webpage with appropriate webcontrols.
- Create a servlet class file
    - Inside the doPost() method :-
    - Copy the required parameters into local variables using getParameter() method of request object
    - Perform JDBC process for select query by specifying the passed userID in the where clause
    - Create a printwriter object using the getWriter() method of response object.
- If any records found for the given userID then return an appropriate error message else return a welcome message to the end user by writing the html code in the servlet class file using the printwriter object.
- Create a JSP page
    - Use the JSTL core library for helping user edit his/her personal details by creating the variables and their values to be modified
    - Use the JSTL SQL library to interact with the database to update the desired user information by performing update operation.
- Return appropriate status to the registered user.

EXPERIMENT – 8

Topic: Custom Tags (Tag extension)

Aim: Create custom tags: date and header. The date tag is used to display current date and header tag is used to print the header in proper format. The header tag has following attributes: align, border, bgcolor, color, font, and size. Show the usage of these two tags in your JSP page. The align, color, font, and size are for alignment of text, color of text, font-family for text, and size of text respectively. The border, and bgcolor are for border size of box containing text, and background color of box respectively.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1, Web Browser

Procedure:

- Create a new WebApplication
  - Write the tag handler class.
    - Output, the current date and the header to the end user, in HTML format in doStartTag() method.
  - Create a tag file.
  - Create the Tag Library Descriptor(TLD) file and specify the tag file and body content(if any).
  - Reference the TLD in web.xml
  - Use the date tag and header tag in your JSP page by specifying the reference to tld file in **taglib uri.**

EXPERIMENT – 9

Topic: Native Methods (JNI)

Aim: Create native the method print to print floating point value right aligned with specified width and precision.

i.e print(55.444,12,2); should generate output containing 5spaces followed by value 55.44. (Total width 12 and 2 digits after decimal point)

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1

Procedure:

- Create a Java class that declares the native method.
- Implement this native method that calls the actual C routine i.e. print() method.
- Compile the java class created in first step.
- Run the java class using **javah** which creates a header file that can be used in C file.
- Code your native method i.e. print( ) in C and include the header file created in the previous step.
- Create a shared library file using the C code by making a DLL.
- Now use the java class like any other ordinary class.

EXPERIMENT – 10

Topic: EJB (Session Bean)

Aim: Create and use a session bean to calculate the income-tax on annual income. The bean takes salary (annual income), and total investment as arguments to business method and returns calculated income-tax as result. The business rules for calculating income-tax are as follows. No income-tax on first 100,000 Rs. of salary. 10% tax on next 100,000 Rs. of remaining salary, 20% on next 100,000 Rs. of remaining salary, 30% on next 100,000 Rs. of remaining salary, and 100% on remaining salary. The investment of maximum Rs.100,000 is considered as non-chargeable income.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1, Web Browser

Procedure:

- Create an new enterprise application
  - Create a appropriate JSP page for your application.
  - Create a java home interface file which contains the calculateInterest() method
  - Define enterprise bean class and implement all the necessary methods along with calculateInterest() method by coding proper logic to calculate the interest.
  - Create a servlet class that will be used to call the session bean.
  - Modify the web.xml file accordingly
  - Deploy the project and run it in your web browser.

EXPERIMENT – 11

Topic: EJB (Entity Bean)

Aim: Create and use CMP entity bean to model Student database table. The student has student ID, name, semester, and roll no. The bean Provides facility of searching a student based on his student ID.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1, Web Browser

Procedure:

- Create an new enterprise application
  - Create a appropriate JSP page for your application
  - Create a java file which will contain the getter and setter methods.
  - Create an enterprise entity bean that will implement the local interface which will contain the business logic for finding the student based on his/her ID.
  - Modify the web.xml file accordingly
  - Deploy the project and run it in your web browser.

EXPERIMENT – 12

(1) Aim: Write a web application for online shopping of books.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1, Web Browser

Procedure:

- Create a new web application project
- Create required web components like JSP, Servlet and static resources(HTML files)
- Create JavaBeans required to implement Business Logic.
- Create Data source and Database.
- Edit all components to implement application logic.
- Build, Deploy and Run the project.

(2) Aim: Develop a javamail application to send and receive emails.

Tools /Apparatus: JDK 1.6 or above , Netbeans IDE 6.1, Web Browser

Procedure:

- Create a web application using NetBeans IDE
- Create Model part of application which consist of components like MailUserBean, TagLibrary, Attachment Servlet.
- Create View part of application
- Create control part of application
- Build and Deploy application