PRACTICALS INDEX

Sr. No.	Title	Page No.
1	Write a program for creating mini chat application using socket programming.	7
2	Write a program for Addition and Subtraction using concept of RMI programming.	11
3	Write a program to implement CRUD operation in JDBC.	18
4	Create Exam Registration Form using JDBC Connectivity.	21
5	Write a program for Creating Edit Menu for Notepad using Frame.	35
6	Write a program for creating simple servlet with JDBC.	45
7	Create Employee information Form using JSP.	69
8	Write a program for implementing concept of MVC Architecture.	92
9	Write a program for implementing concept of Hibernate, Stuct, Spring.	103

10	Write a program for implementing concept of Maven Project.	118
11	Write a program for implementing concept of Web Service.	122
12	Write a program for implementing concept of Junit Service.	123
13	Write a program for implementing concept of JAX.	133

Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 1

Aim: Write a program for creating mini chat application using socket programming.

Theory: Java Socket Programming

Java Socket programming is used for communication between the applications running on different JRE.

Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:
IP Address of Server, and
Port number.
Here, we are going to make one-way client and server communication. In this application, client sends a message to the server, server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept() method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

Socket class

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Important methods

Method	Description
1) public InputStream getInputStream()	returns the InputStream attached with this socket.
2) public OutputStream getOutputStream()	returns the OutputStream attached with this socket.
3) public synchronized void close()	closes this socket

ServerSocket class

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Important methods

Method	Description
1) public Socket accept()	returns the socket and establish a connection between server and client.
2) public synchronized void close()	closes the server socket.

Example of Java Socket Programming

Creating Server:

To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection and waits for the client

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

Socket s=new Socket("localhost",6666);

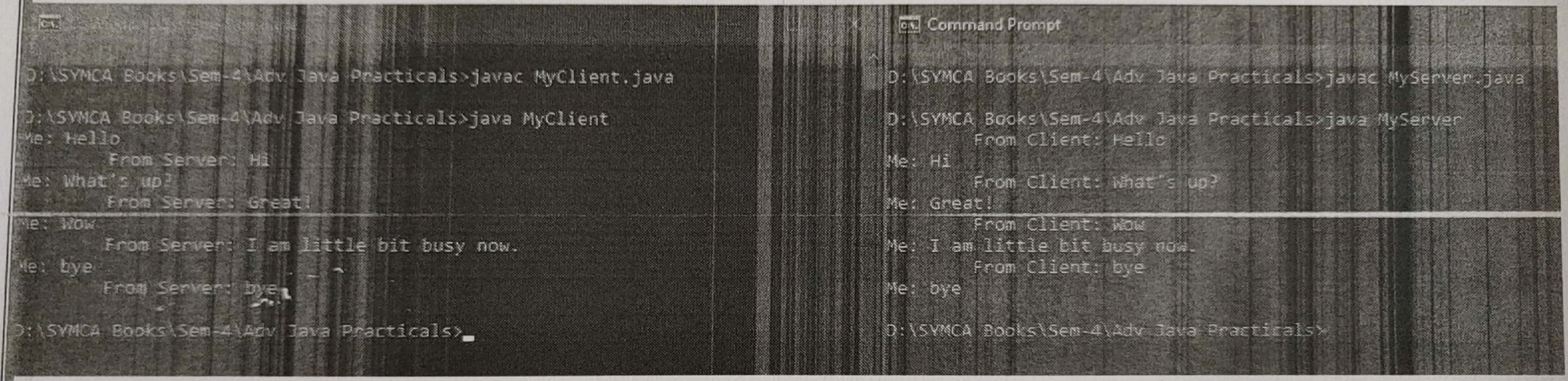
MyServer.java

```
import java.io.*;
import
j    java.net.*;
public class MyServer{
public static void main(String[] ar){ try{
ServerSocket ss = new ServerSocket(1234); Socket s = ss.accept();

DataInputStream din = new DataInputStream(s.getInputStream()); DataOutputStream dout =
new
DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new   InputStreamReader(System.in));
StringfromClient="",toClient="";while(!fromClient.equals(
"bye")){
fromClient= (String) din.readUTF(); System.out.println("\tFrom Client:     "
+fromClient); System.out.print("Me: ");
toClient=br.readLine();
dout.writeUTF(toClient); dout.flush();
}
din.close();
s.close();
ss.close();
}catch(Exception e){
System.out.println(e);
    }
  }
 }
```

MyClient.java

```
import java.io.*;

import java.net.*;

public class MyClient{

public static void main(String[] ar){ try{
Socket s = new Socket("localhost",1234);
DataInputStream din = new DataInputStream(s.getInputStream()); DataOutputStream dout
= new
DataOutputStream(s.getOutputStream());BufferedReaderbr=newBufferedReader(new
InputStreamReader(System.in));
StrinfromServer="",toServer="";
while(!fromServer.equals("bye")){
System.out.print("Mc: toServer=br.readLine();
dout.writeUTF(toServer); dout.flush();
fromServer= (String)din.readUTF(); System.out.println("\tFrom Server: "
+fromServer);
}
dout.close();
s.close()
}catch(Exception e){
System.out.println(e);
                    }
                }
        }
```

-: Output



Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 2

Aim : Write a program for Addition and Subtraction using concept of RMI programming

Theory: RMI (Remote Method Invocation)

The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects stub and skeleton.

Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.

A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

It initiates a connection with remote Virtual Machine (JVM)
It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
It waits for the result
It reads (unmarshals) the return value or exception, and
It finally, returns the value to the caller.

skeleton

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

It reads the parameter for the remote method
It invokes the method on the actual remote object, and
It writes and transmits (marshals) the result to the caller.

In the Java 2 SDK, an stub protocol was introduced that eliminates the need for skeletons.

RMI Example

The is given the 6 steps to write the RMI program.

Create the remote interface

Provide the implementation of the remote interface
Compile the implementation class and create the stub and skeleton objects using the rmic tool
Start the registry service by rmiregistry tool

Create and start the remote application
Create and start the client application
create the remote interface
For creating the remote interface, extend the Remote interface and declare the RemoteException with all the methods of the remote interface. Here, we are creating a remote interface that extends the Remote interface. There is only one method named add() and it declares RemoteException.

```
import java.rmi.*;
public interface Adder extends Remote{
public int add(int x,int y)throws RemoteException;
}
```

Provide the implementation of the remote interface
Now provide the implementation of the remote interface. For providing the implementation of the Remote interface, we need to
Either extend the UnicastRemoteObject class,
or use the exportObject() method of the UnicastRemoteObject class
In case, you extend the UnicastRemoteObject class, you must define a constructor that declares RemoteException.

```
import java.rmi.*;
import java.rmi.server.*;
public class AdderRemote extends UnicastRemoteObject implements Adder{
AdderRemote()throws RemoteException{
super();
}
public int add(int x,int y){return x+y;}
}
```

create the stub and skeleton objects using the rmic tool.
Next step is to create stub and skeleton objects using the rmi compiler. The rmic tool invokes the RMI compiler and creates stub and skeleton objects.

rmic AdderRemote
Start the registry service by the rmiregistry tool
Now start the registry service by using the rmiregistry tool. If you don't specify the port number, it uses a default port number. In this example, we are using the port number 5000.

rmiregistry 5000
Create and run the server application
Now rmi services need to be hosted in a server process. The Naming class provides methods to get and store the remote object. The Naming class provides 5 methods.

public static java.rmi.Remote lookup(java.lang.String) throws java.rmi.NotBoundException, java.net.MalformedURLException, java.rmi.RemoteException; It returns the reference of the remote object.
public static void bind(java.lang.String, java.rmi.Remote) throws java.rmi.AlreadyBoundException, java.net.MalformedURLException, java.rmi.RemoteException; It binds the remote object with the given name.
public static void unbind(java.lang.String) throws java.rmi.RemoteException, java.rmi.NotBoundException,

java.net.MalformedURLException; It destroys the remote object which is bound with the given name.
public static void rebind(java.lang.String, java.rmi.Remote) throws java.rmi.RemoteException, java.net.MalformedURLException; It binds the remote object to the new name.
public static java.lang.String[] list(java.lang.String) throws java.rmi.RemoteException, java.net.MalformedURLException; It returns an array of the names of the remote objects bound in the registry.
In this example, we are binding the remote object by the name sonoo.

```
import java.rmi.*;
import java.rmi.registry.*;
public class MyServer{
public static void main(String args[]){
try{
Adder stub=new AdderRemote();
Naming.rebind("rmi://localhost:5000/sonoo",stub);
}catch(Exception e){System.out.println(e);}
}
}
```

Create and run the client application

At the client we are getting the stub object by the lookup() method of the Naming class and invoking the method on this object. In this example, we are running the server and client applications, in the same machine so we are using localhost. If you want to access the remote object from another machine, change the localhost to the host name (or IP address) where the remote object is located.

```
import java.rmi.*;
public class MyClient{
public static void main(String args[]){
try{
Adder stub=(Adder)Naming.lookup("rmi://localhost:5000/sonoo");
System.out.println(stub.add(34,4));
}catch(Exception e){}
}
}
```

download this example of rmi

For running this rmi example,

compile all the java files

javac *.java

2)create stub and skeleton object by rmic tool

rmic AdderRemote

3)start rmi registry in one command prompt

rmiregistry 5000

4)start the server in another command prompt

java MyServer

5)start the client application in another command prompt

java MyClient

PROGRAM :

Calculator.java

```
import java.rmi.*;
public interface Calculator extends Remote {
    public int add(int a,int b)throws
RemoteException; public int
    sub(inta, int b) throws RemoteException;
}
```

RemoteCalculator.java

```
import java.rmi.*;
import
java.rmi.server.*;
public class RemoteCalculator extends UnicastRemoteObject implements
Calculator{ RemoteCalculator()throws RemoteException{
            super();
        }
        public int add(int a,int
        b){return a+b;} public int
        sub(int a,int b){return a-b;}
}
```

MyServer1.java

```
import java.rmi.*;
import
java.rmi.registry.*;
public class
MyServer1{
        public static void main(String
        args[]){ try{

                        Calculator stub=new RemoteCalculator();
                        Naming.rebind("rmi://localhost:1234/sameer",
                        stub);
                }catch(Exception e){System.out.println(e);}
        }
}
```

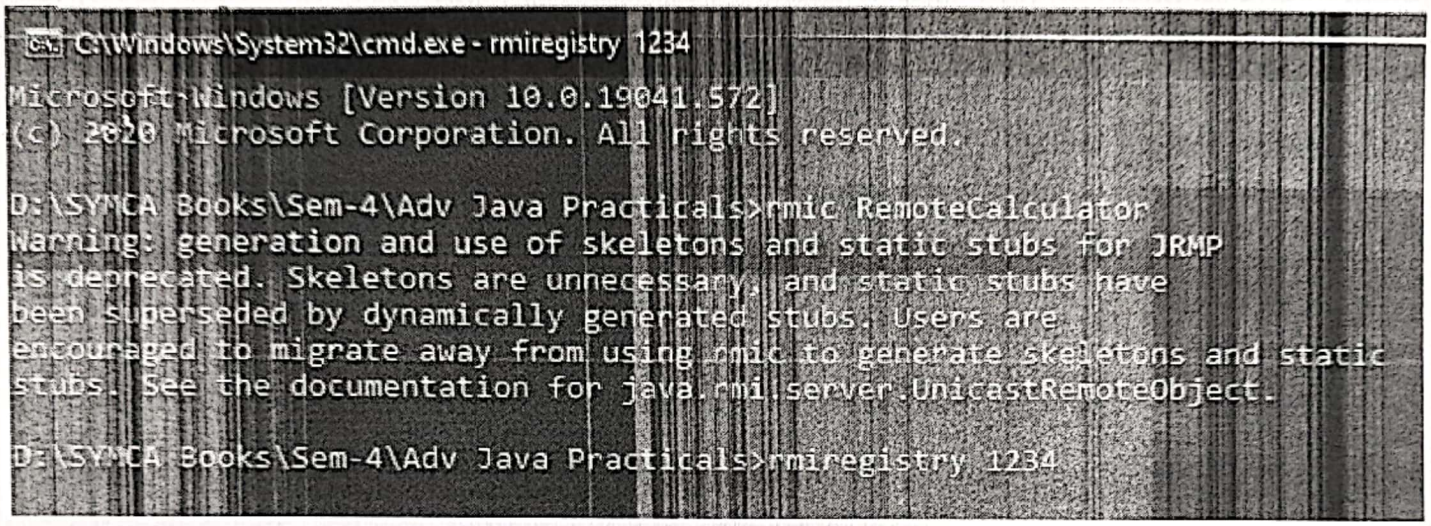
MyClient1.java

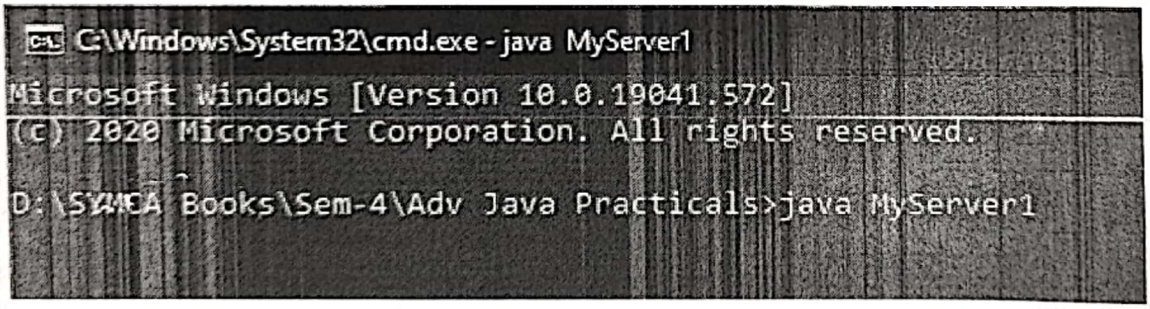
```
import java.rmi.*;
import
java.util.Scanner;
public class
MyClient1{
        public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int no1, no2;
                System.out.print("Enter two integer
                numbers : "); no1 = sc.nextInt();
                no2 =
                sc.nextInt()
                ; try{
                        Calculator
                        stub=(Calculator)Naming.lookup("r
                        mi://localhost:1234/sameer");
                        System.out.println("Addition : "+no1+" +
                        "+no2+"                =
                        "+stub.add(no1,no2));
                        System.out.println("Subtraction : "+no1+" - "+no2+" =
"+stub.sub(no1,no2));
                }catch(Exception e){System.out.print(e);} } }
```

-:

Output

:-





Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 3

Aim: : Write a program to implement CRUD operation in JDBC.

Theory:

CRUD in Servlet

A CRUD (Create, Read, Update and Delete) application is the most important application for any project development. In Servlet, we can easily create CRUD application.

Servlet CRUD example

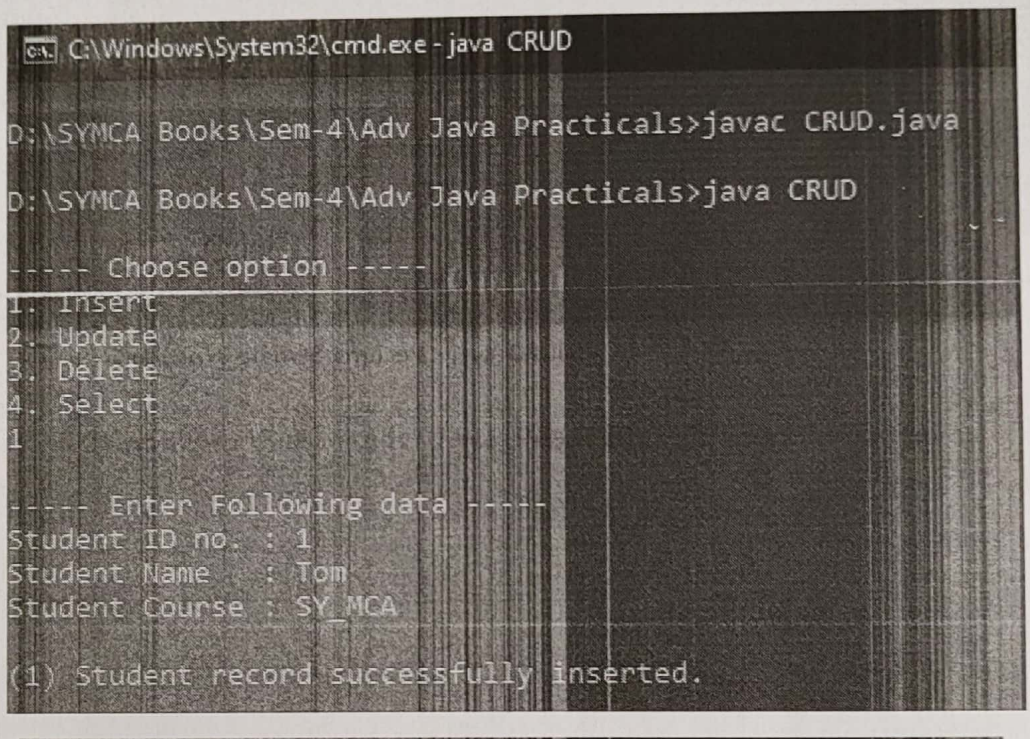
Create "user905" table in Oracle Database with auto incrementing id using sequence. There are 5 fields in it: id, name, password, email and country.

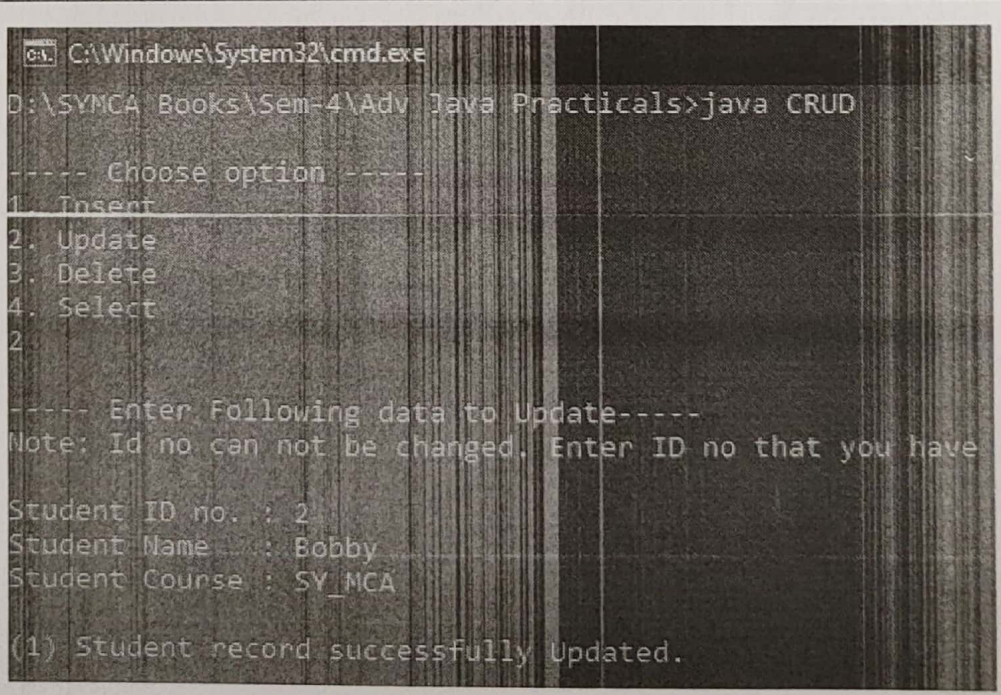
Program :

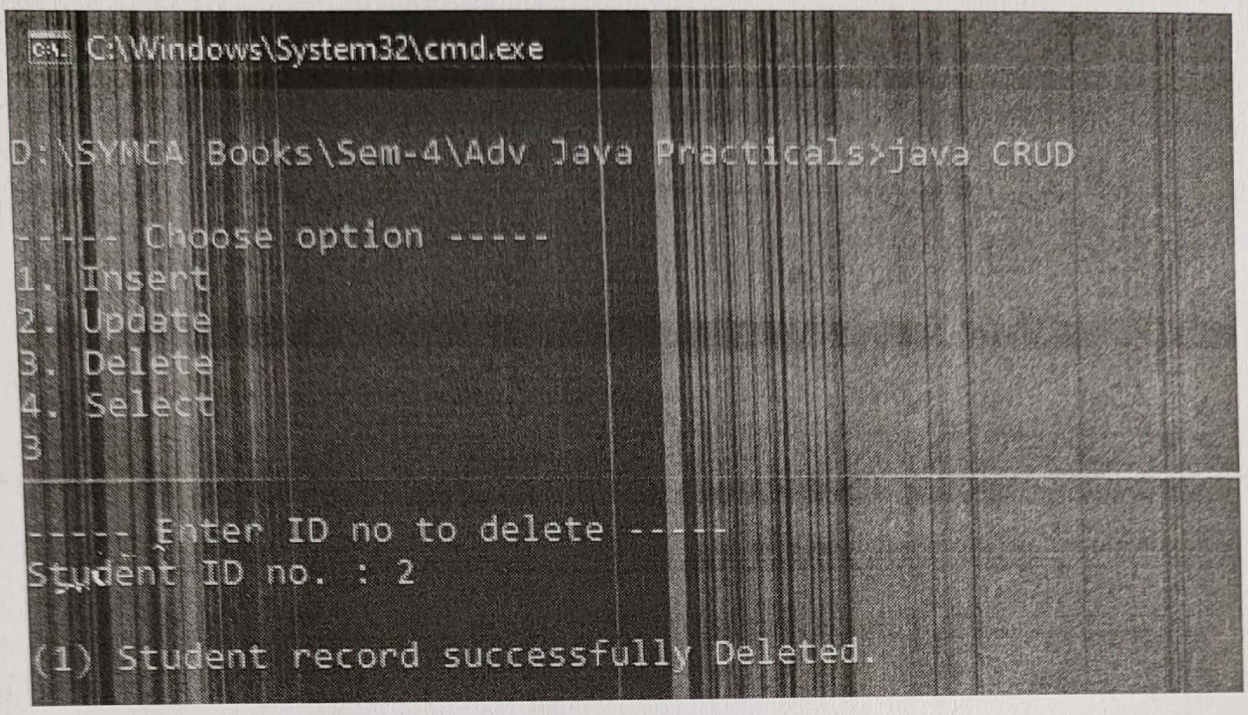
```
 import java.sql.*;
import java.util.Scanner;
class CRUD
{
  Statement stmt;
Connection con;
        public int insert(String id, String name, String course) throws SQLException
        {
                return stmt.executeUpdate("INSERT INTO
StudentData VALUES('"+id+"','"+name+"','"+course+"')");
        }
        public int update(String id, String name, String course) throws SQLException
        {
                return stmt.executeUpdate("UPDATE StudentData SET
name='"+name+"', course='"+course+"' WHERE id = '"+id+"'");
        }
        public int delete(String id) throws SQLException
        {
                return stmt.executeUpdate("DELETE FROM StudentData WHERE id =
'"+id+"'");
        }
        public void select() throws SQLException
        {
                ResultSet rs=stmt.executeQuery("select * from StudentData order by(id)");
                System.out.println("------------------------ ");
                System.out.println("S_id \tS_name \tS_course");
                System.out.println("------------------------ ");
                while(rs.next())
                System.out.println(rs.getInt(1)+" \t"+rs.getString(2)+"\t"+rs.getString(3));
        }
```

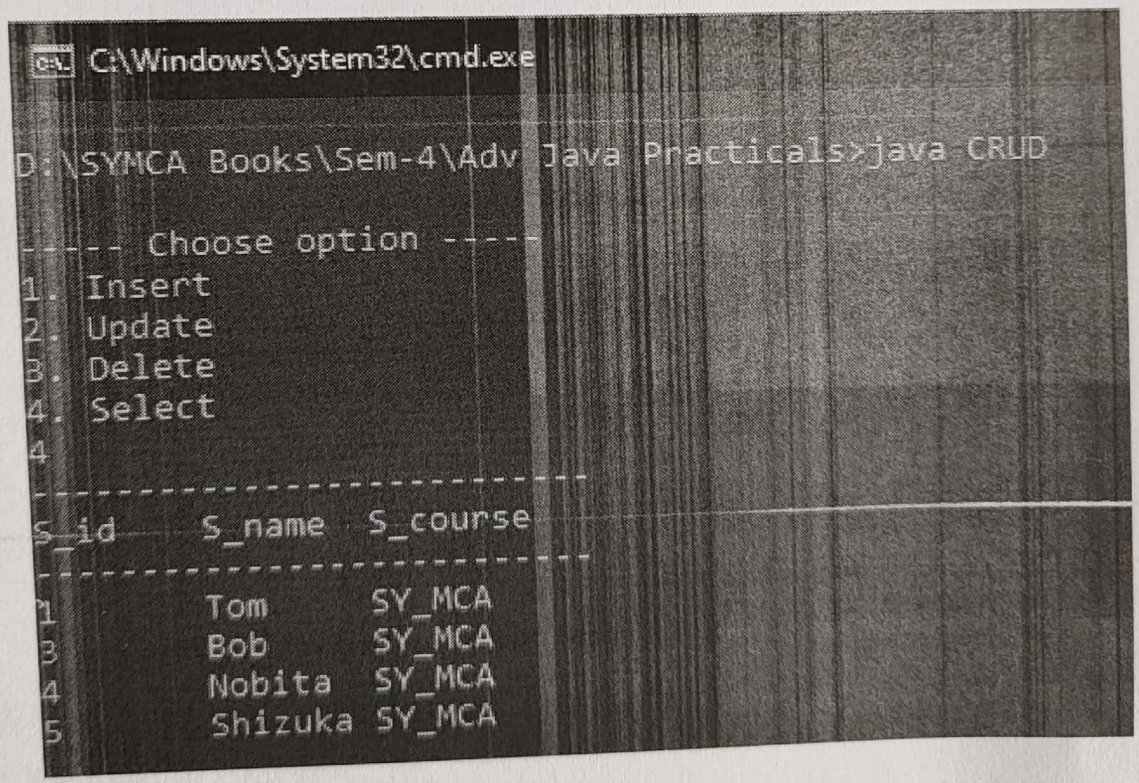
```
public static void main(String args[])
        {
Scanner sc = new Scanner(System.in);
CRUD crud = new CRUD();
int choice;
String id,name,cours
```

-: Output :-









Class: MCA

Subject: Lab I- Advance Java

Experiment No. 4

Aim: Create Exam Registration Form using JDBC Connectivity.

Theory:

Example of Registration form in servlet

Here, you will learn that how to create simple registration form in servlet. We are using oracle10g database. So you need to create a table first as given below:

```
CREATE TABLE  "REGISTERUSER"
  (    "NAME" VARCHAR2(4000),
   "PASS" VARCHAR2(4000),
   "EMAIL" VARCHAR2(4000),
   "COUNTRY" VARCHAR2(4000)
  )
 hn =0p;[
```

To create the registration page in servlet, we can separate the database logic from the servlet. But here, we are mixing the database logic in the servlet only for simplicity of the program

Example of Registration form in servlet

In this example, we have created the three pages.

register.html

Register.java

web.xml

register.html

In this page, we have getting input from the user using text fields and combobox. The information entered by the user is forwarded to Register servlet, which is responsible to store the data into the database.

Register.java

This servlet class receives all the data entered by user and stores it into the database. Here, we are performing the database logic. But you may separate it, which will be better for the web application.

web.xml file

The is the configuration file, providing information about the servlet.

Prgram :

Index.html

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
 <title>Exam Registration</title>
  <style>
   body{
   font-family:Calibri;
background-color: #fff5e6;
      }
      .container       {
        padding: 50px;
        background-col    #ffc266;
        padding-left:        100px;
        padding-right:       100px;
        margin-left:100px;
        margin-right: 100px;
        }

       input[type=text], input[type=password], textarea {
         width: 100%;
         padding:          15px;
         margin:  5px  0  22px
         0;

         display :inline-block;
         border: none;
         background: #fff;
       }
```

```
    input[type=text]:focus,
    input[type=password]:focus { background-color:
    #ffe680;
    outline: none;
}
div {
    padding: 10px 0;
}
hr {
    border: 1px solid
    #f1f1f1; margin-bottom:
    25px;
}
.registerbtn {
    background-color:
    #00D0FF; color: white;
    padding: 16px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 100%;
    opacity: 0.9;
}
.registerbtn:hover {
    opacity: 1;
}
table {
    background: #fff;
    padding: 10px 20px;
```

```
    width: 100%;
    margin: 5px 0 22px 0;
  }
  select:focus {
  background-color: #ffe680; outline: none;
   }
</style>
<script type="text/javascript"> function
   validateform() {
      var id = document.register.id.value;
      var course = document.register.Course.value; var mobile =
      document.register.mobileno.value; var email =
      document.register.email.value;
      var atposition = email.indexOf("@"); var dotposition =
      email.lastIndexOf("."); if (isNaN(id)) {
         alert("Invaid ID"); return false;
      }
      if (course == "Course") { alert("Please select
         Course"); return false;
      }
      if (isNaN(mobile)) {
         alert("Invalid Mobile Number"); return false;
      }
      if (mobile.length != 10) {
         alert("Invalid Mobile Number\nInsert number without (+91)"); return false;
      }
```

```
        var mailformat = /^[a-zA-Z0-9.!#$%&'*+/=?^_`{|}~-]+@[a-zA-Z0-9-]+(?:\.[a-zA-
        Z0-9-
        ]+)*$/;

        if (email.match(mailformat))
          return true;
        else
        {
          alert("Invalid email address!");
          return false;
        }
      }
    </script>
  </head>
  <body>
    <form name = "register" action="Register" method="get" onsubmit="return
validateform()">
      <div class="container">
        <center> <h1 style="color:white;"> Student Registration Form</h1> </center>
        <hr>
        <label> ID Number: </label>
        <input type="text" name="id" placeholder= "ID Number" size="15" required />
        <label> First name: </label>
        <input type="text" name="firstname" placeholder= "Firstname" size="15" required
/>
        <label> Middle name: </label>
        <input type="text" name="middlename" placeholder="Middlename" size="15"
required />
        <label> Last name: </label>
        <input type="text" name="lastname" placeholder="Lastname" size="15" required
/>
        <div>
          <table>
```

```
<tr>
  <td>
    <label>
      Course :
    </label>

    <select name="Course">
      <option value="Course">Course</option>
      <option value="BCA">BCA</option>
      <option value="BBA">BBA</option>
      <option value="B.Tech">B.Tech</option>
      <option value="MBA">MBA</option>
      <option value="MCA">MCA</option>
      <option value="M.Tech">M.Tech</option>
    </select>
  </td>
  <td>
    <label>
      Semester :
    </label>

    <select name="Sem">
      <option value="I">I</option>
      <option value="II">II</option>
      <option value="III">III</option>
      <option value="IV">IV</option>
      <option value="V">V</option>
      <option value="VI">VI</option>
    </select>
  </td></tr></table>
```

```
        </div>
        <div>
          <table>
            <tr>
              <td><label>
                  Gender :
                </label></td>
              <td><input type="radio" value="Male" name="gender" checked > Male</td>
              <td><input type="radio" value="Female" name="gender"> Female</td>
              <td><input type="radio" value="Other" name="gender"> Other</td>
            </tr>
          </table>
        </div>
        <label>
          Mobile No.:
        </label>
        <input type="text" name="mobileno" placeholder="Mobile no." size="10"
required>
        Address :
        <textarea cols="80" rows="5" name="address" >
        </textarea>
        <label for="email">Email</label>
        <input type="text" placeholder="Enter Email" name="email" required>
        <button type="submit" class="registerbtn">Register</button>
    </form>
  </body>
</html>
```

Database –> tblStudentExam table schema:

```
Create table tblStudentExam
(
        id int primary key,
        fname varchar(20),
        mname varchar(20),
        lname varchar(20),
        course varchar(10),
        sem varchar(5),
        gender varchar(10),
        mobile varchar(12),
        address varchar(100),
        email varchar(50)
);
```

DatabaseConn.java

```
package myPack;
import java.sql.*;

public class DatabaseConn {
  public String dbConn(int id, String fname, String mname, String lname, String
course, String sem,
      String gender, String mobile, String address, String email)
    { String msg = null;
    try {
      Connection con = null;
      Class.forName("oracle.jdbc.driver.OracleDriver")
      ;
      con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"Sameer", "Sameer");
      System.out.println("Connection prepared");
      PreparedStatement ps = con.prepareStatement(
          "insert into tblStudentExam values(?,?,?,?,?,?,?,?,?,?)");
```

```
            ps.setInt(1, id);
            ps.setString(2, fname);
            ps.setString(3, mname);
            ps.setString(4, lname);
            ps.setString(5, course);
            ps.setString(6, sem);
            ps.setString(7, gender);
            ps.setString(8, mobile);
            ps.setString(9, address);
            ps.setString(10, email);
            int i = ps.executeUpdate();
            con.close();
            if (i > 0) {
               msg = "You are successfully registered...";
            } else {
               msg = "Something went wrong...! Please try again or contact admin";
            }

        } catch (ClassNotFoundException | SQLException e)
           { System.err.print(e);
        }
        return msg;
    }
}
```

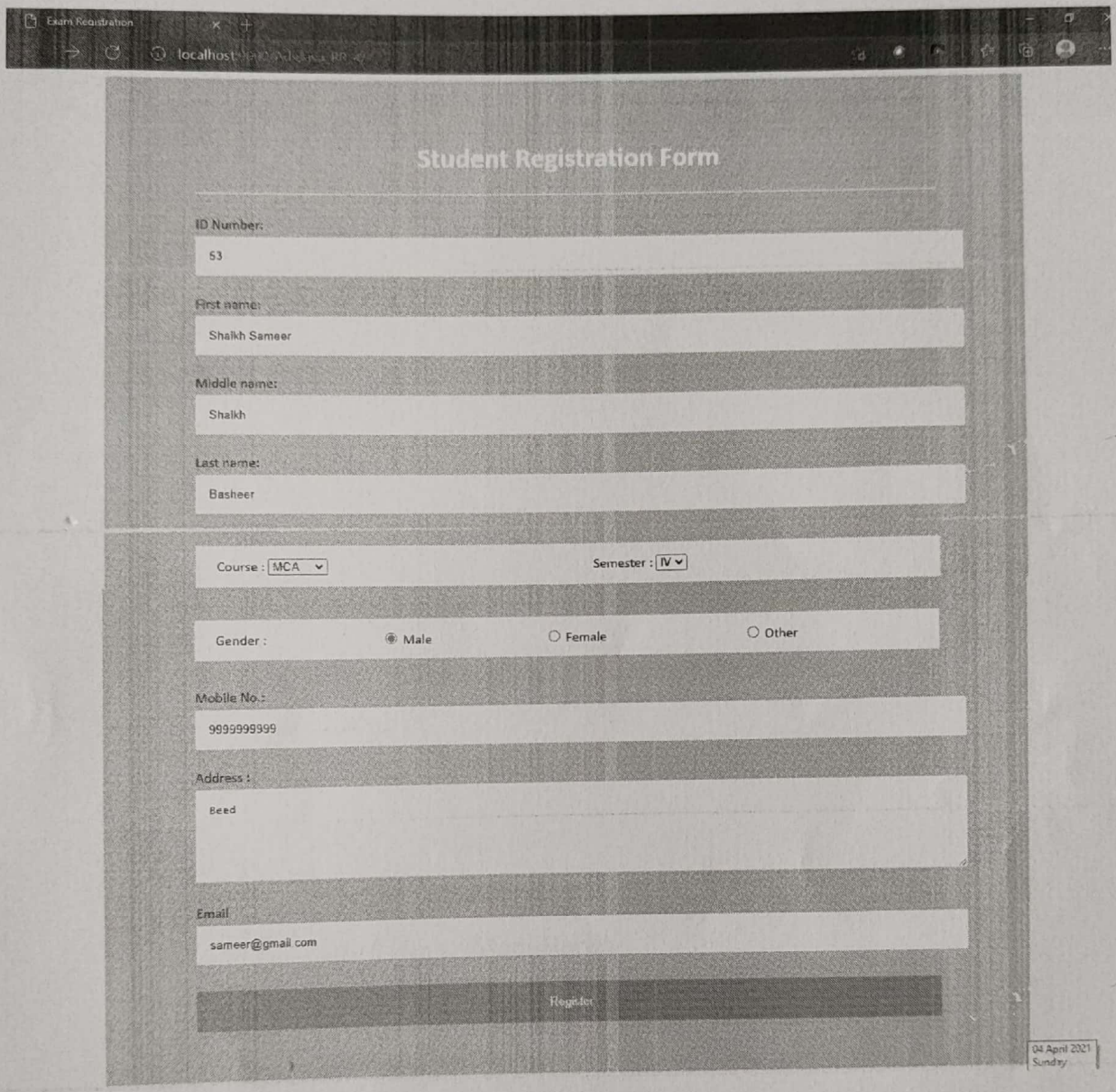
Register.java (Servlet)

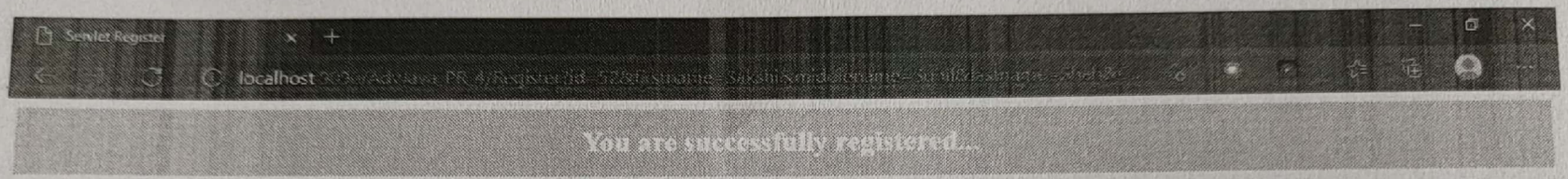
```
package myPack;

import
java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet(name = "Register", urlPatterns =
{"/Register"}) public class Register extends HttpServlet {

  protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
      throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
      /* TODO output your page here. You may use following sample code. */
      out.println("<!DOCTYPE html>");
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Servlet Register</title>");
      out.println("</head>");
      out.println("<body bgcolor='#fff5e6'>");

      String fname, mname, lname, course, sem, gender, mobile, address, email;
      int id = 0;
```

```
try {
    id = Integer.parseInt(request.getParameter("id"));
    fname = request.getParameter("firstname");
    mname = request.getParameter("middlename");
    lname = request.getParameter("lastname"); course
    = request.getParameter("Course");
    sem = request.getParameter("Sem");
    gender = request.getParameter("gender");
    mobile = request.getParameter("mobileno");
    address = request.getParameter("address");
    email = request.getParameter("email");

    DatabaseConn dbc = new DatabaseConn();
    String msg = dbc.dbConn(id, fname, mname, lname, course, sem, gender, mobile,
address, email);
    out.println("<style>");
    out.println("h2{");
    out.println("background-color: #ffc266;");
    out.println("padding: 15px;");
    out.println("margin: 5px 0 22px 0;");
    out.println("color:white;");
    out.println("}");
    out.println("</style>");

    if (msg == null) {
      out.println("<h2 align='center'>Already registered.</h2>");
    } else {
      out.println("<h2 align='center'>" + msg + "</h2>");
    }
      out.println("</body>");
```

```
            out.println("</html>");
        } catch (Exception e) {
            PrintWriter pw = response.getWriter();
            pw.println("Exception : " + e);
        }
    }
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
  processRequest(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
  processRequest(request, response);
}

@Override
public String getServletInfo()
  { return "Short description";
}
}
```

-: Output :-





Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 5

Aim: Write a program for Creating Edit Menu for Notepad using Frame.

Program :

```
public class Notepad extends javax.swing.JFrame {

  public Notepad() {
  initComponents();
  }

  @SuppressWarnings("unchecked")
  // <editor-fold defaultstate="collapsed" desc="Generated Code">
  private void initComponents() {

    jScrollPane1 = new
    javax.swing.JScrollPane(); jEditorPane1 =
    new javax.swing.JEditorPane(); jMenuBar1 =
    new javax.swing.JMenuBar(); jMenu1 = new
    javax.swing.JMenu(); jMenuItem2 = new
    javax.swing.JMenuItem(); jMenu2 = new
    javax.swing.JMenu(); jMenuItem3 = new
    javax.swing.JMenuItem(); jMenuItem4 = new
    javax.swing.JMenuItem(); jMenuItem5 = new
    javax.swing.JMenuItem(); jMenuItem6 = new
    javax.swing.JMenuItem(); jMenu3 = new
    javax.swing.JMenu(); jMenuItem7 = new
    javax.swing.JMenuItem();



    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOS
```

```
E); setTitle("My Notepad");
setFont(new java.awt.Font("Arial", 1, 14)); // NOI18N

jEditorPane1.setFont(new java.awt.Font("Segoe UI", 0, 18)); //
NOI18N jScrollPane1.setViewportView(jEditorPane1);

jMenuBar1.setBorder(null);
jMenuBar1.setFont(new java.awt.Font("Consolas", 0, 24)); // NOI18N
jMenuBar1.setPreferredSize(new java.awt.Dimension(66, 30));

jMenu1.setText("File");
jMenu1.setFont(new java.awt.Font("Segoe UI", 0, 16)); // NOI18N


jMenuItem2.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.
V K_F4, java.awt.event.InputEvent.ALT_MASK));
jMenuItem2.setFont(new java.awt.Font("Segoe UI", 0, 16)); // NOI18N
jMenuItem2.setText("Exit");
jMenuItem2.addActionListener(new java.awt.event.ActionListener()
  { public void actionPerformed(java.awt.event.ActionEvent evt) {
    jMenuItem2ActionPerformed(evt);
  }
});
jMenu1.add(jMenuItem2);

jMenuBar1.add(jMenu1);

jMenu2.setText("Edit");
jMenu2.setFont(new java.awt.Font("Segoe UI", 0, 16)); // NOI18N
```

```
<tr>
  <td colspan="4">
    <table border='1' cellspacing="0" width="100%" bgcolor="#98FB98">
      <tr>
        <th>Subject</th>
        <th>Theory</th>
        <th>CT</th>
        <th>TA</th>
      </tr>
      <tr>
        <td>
          <select name="subject1">
            <option value="Advanced Java" selected>Advanced Java</option>
            <option value="Distributed Databases">Distributed
Databases</option>
            <option value="Software Testing Techniques">Software Testing
Techniques</option>
            <option value="Cloud Computing">Cloud Computing</option>
            <option value="Pytmming">Python
Programming</option>
```

```
                </select>
              </td>
              <td>
                <input type="text" name="t1" placeholder= "Marks" required />
              </td>
              <td>
                <input type="text" name="c1" placeholder= "Marks" required />
              </td>
              <td>
                <input type="text" name="TA1" placeholder= "Marks" required />
            </td></tr>
            <tr>
              <td>
                <select name="subject2">
                  <option value="Advanced Java">Advanced Java</option>
                  <option value="Distributed Databases" selected>Distributed
Databases</option>
                  <option value="Software Testing Techniques">Software Testing
Techniques</option>
                  <option value="Cloud Computing">Cloud Computing</option>
                  <option value="Python Programming">Python
Programming</option>
                </select>
              </td>
              <td>
                <input type="text" name="t2" placeholder= "Marks" required />
              </td>
              <td>
                <input type="text" name="c2" placeholder= "Marks" required />
              </td>
              <td>
                <input type="text" name="TA2" placeholder= "Marks" required />
              </td>
            </tr>
```

```
<tr>
  <td>
    <select name="subject3">
      <option value="Advanced Java">Advanced Java</option>
      <option value="Distributed Databases">Distributed
Databases</option>
      <option value="Software Testing Techniques" selected>Software
Testing Techniques</option>
      <option value="Cloud Computing">Cloud Computing</option>

<option value="Python Programming">Python
    Programming</option>
    </select>
  </td>
  <td>
    <input type="text" name="t3" placeholder= "Marks" required />
  </td>
  <td>
    <input type="text" name="c3" placeholder= "Marks" required />
  </td>
  <td>
    <input type="text" name="TA3" placeholder= "Marks" required />
  </td>
</tr><tr>
  <td>
    <select name="subject4">
      <option value="Advanced Java">Advanced Java</option>
      <option value="Distributed Databases">Distributed
Databases</option>
      <option value="Software Testing Techniques">Software Testing
Techniques</option>
      <option value="Cloud Computing" selected>Cloud
Computing</option>
      <option value="Python Programming">Python
Programming</option>
    </select>
```

```
</td>
<td>
<input type="text" name="t4" placeholder= "Marks" required />
</td>
<td>
<input type="text" name="c4" placeholder= "Marks" required />
</td>
<td>
<input type="text" name="TA4" placeholder= "Marks" required />
</td>
</tr>
<tr>
<td>
<select name="subject5">
<option value="Advanced Java">Advanced Java</option>
<option value="Distributed Databases">Distributed Databases</option>
<option value="Software Testing Techniques">Software Testing Techniques</option>
<option value="Cloud Computing">Cloud Computing</option>
<option value="Python Programming" selected >Python Programming</option>
</select>
</td>
<td>
<input type="text" name="t5" placeholder= "Marks" required />
</td>
<td>
<input type="text" name="c5" placeholder= "Marks" required />
</td>
<td>
<input type="text" name="TA5" placeholder= "Marks" required />
</td>
</tr>
</table>
<table border='1' cellspacing="0" width="100%"
```

```
bgcolor="#98FB98">
<label style="align:left; font-weight: bold; font-size: 20px;">Lab:</label>
<tr>
  <th>Subject</th>
  <th>Internal</th>
  <th>External</th>
</tr>
<tr>
  <td>
    <select name="subject6">
      <option value="Lab: Advanced Java" selected>Lab: Advanced Java</option>
      <option value="Lab:Software Testing Techniques">Lab:Software Testing Techniques</option>
      <option value="Lab: Cloud Computing">Lab: Cloud Computing</option>
      <option value="Lab:Python Programming">Lab:Python Programming</option>
    </select>
  </td>
  <td>
    <input type="text" name="i1" id="marks" placeholder= "Marks" required />
  </td>
  <td>
    <input type="text" name="e1" id="marks" placeholder= "Marks" required />
  </td>
</tr>
<tr>
  <td>
    <select name="subject7">
      <option value="Lab: Advanced Java">Lab: Advanced Java</option>
```

```
                    <td>
                      <input type="text" name="i2" placeholder= "Marks" required />
                    </td>
                    <td>
                      <input type="text" name="e2" placeholder= "Marks" required />
                    </td>
                  </tr>
                  <tr>
                    <td>
                      <select name="subject8">
                        <option value="Lab: Advanced Java">Lab: Advanced Java</option>
                        <option value="Lab:Software Testing
Techniques">Lab:Software Testing Techniques</option>
                        <option value="Lab: Cloud Computing" selected>Lab: Cloud
Computing</option>

Programming</option>
                        <option value="Lab:Software Testing
Techniques" selected>Lab:Software Testing Techniques</option>
                        <option value="Lab: Cloud Computing">Lab: Cloud
Computing</option>
                        <option value="Lab:Python Programming">Lab:Python
Programming</option>
                      <
                      / <option value="Lab:Python Programming">Lab:Python
```

```
                    </select>
                  </td>
                  <td>
                    <input type="text" name="i3" placeholder= "Marks" required />
                  </td>
                  <td>
                    <input type="text" name="e3" placeholder= "Marks" required />
                  </td>
                </tr>
                <tr>
                  <td>
                    <select name="subject9">
                      <option value="Lab: Advanced Java">Lab: Advanced Java</option>
                      <option value="Lab:Software Testing
Techniques">Lab:Software Testing Techniques</option>
                      <option value="Lab: Cloud Computing">Lab: Cloud
Computing</option>
                      <option value="Lab:Python Programming" selected>Lab:Python
Programming</option>
                    </select>
                  </td>
                  <td>
                    <input type="text" name="i4" placeholder= "Marks" required />
                  </td>
                  <td>
                    <input type="text" name="e4" placeholder= "Marks" required />
                  </td>
                </tr>

              </table>
            </td>
          </tr>
          <tr>
            <td colspan="4"><button type="submit"
class="registerbtn">Submit</button></td>
```

```
            </tr>
          </table>
        </form>
    </body>
</html>
```

MyServlet.java (Servlet):

```
package servlet;

import
java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.sql.*;

public class MyServlet extends HttpServlet {

  protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
      out.println("<!DOCTYPE html>");
      out.println("<html>");
      out.println("<head>");
      out.println("<title>Servlet Register</title>");
      out.println("</head>");
      out.println("<body
      bgcolor='#8EE4AF'>");
```

```
out.println("<style>");
out.println("h2{");
out.println("background-color: #5CDB95;"); out.println("padding:
15px;"); out.println("margin: 5px 0 22px
0;"); out.println("color:white;");
out.println("}");
out.println("</style>");

String name, course, class1, sem;
String s1, s2, s3, s4, s5, s6, s7, s8,
s9;
double t1, ct1, ta1, t2, ct2, ta2, t3, ct3, ta3, t4, ct4, ta4, t5, ct5, ta5, internal1,
external1, ta6, internal2, external2, ta7, internal3, external3, ta8, internal4, external4, ta9;
int id;

id = Integer.parseInt(request.getParameter("id"));
name = request.getParameter("fullname"); course
= "MCA";
class1 = request.getParameter("class");
sem = request.getParameter("Sem"); s1
= request.getParameter("subject1");
t1 = Double.parseDouble(request.getParameter("t1")); ct1
= Double.parseDouble(request.getParameter("c1")); ta1 =
Double.parseDouble(request.getParameter("TA1"));

s2 = request.getParameter("subject2");
t2 = Double.parseDouble(request.getParameter("t2")); ct2
= Double.parseDouble(request.getParameter("c2")); ta2 =
Double.parseDouble(request.getParameter("TA2"));

s3 = request.getParameter("subject3");
```

```
t3 = Double.parseDouble(request.getParameter("t3")); ct3
= Double.parseDouble(request.getParameter("c3")); ta3 =
Double.parseDouble(request.getParameter("TA3"));

s4 = request.getParameter("subject4");
t4 = Double.parseDouble(request.getParameter("t4"));
ct4 = Double.parseDouble(request.getParameter("c4")); ta4
= Double.parseDouble(request.getParameter("TA4"));

s5 = request.getParameter("subject5");
t5 = Double.parseDouble(request.getParameter("t5")); ct5
= Double.parseDouble(request.getParameter("c5")); ta5 =
Double.parseDouble(request.getParameter("TA5"));

s6 = request.getParameter("subject6");
internal1 = Double.parseDouble(request.getParameter("i1"));
external1 = Double.parseDouble(request.getParameter("e1"));

s7 = request.getParameter("subject7");
internal2 = Double.parseDouble(request.getParameter("i2"));
external2 = Double.parseDouble(request.getParameter("e2"));

s8 = request.getParameter("subject8");
internal3 = Double.parseDouble(request.getParameter("i3"));
external3 = Double.parseDouble(request.getParameter("e3"));

s9 = request.getParameter("subject9");
internal4 = Double.parseDouble(request.getParameter("i4"));
external4 = Double.parseDouble(request.getParameter("e4"));
```

```
out.println("</h3>");
try {
    Connection connection = null;
    Class.forName("oracle.jdbc.driver.OracleDriver")
    ;
    connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe",
"Sameer", "Sameer");
    String sql = "BEGIN"
        + "
inserttblResult(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);"
        + "END;";

    PreparedStatement ps = connection.prepareStatement(sql);
    ps.setInt(1, id);
    ps.setString(2, name);
    ps.setString(3, course);
    ps.setString(4, class1);
    ps.setString(5, sem);
    ps.setString(6, s1);
    ps.setString(7, s2);
    ps.setString(8, s3);
    ps.setString(9, s4);
    ps.setString(10, s5);
    ps.setString(11, s6);
    ps.setString(12, s7);
    ps.setString(13, s8);
    ps.setString(14, s9);

    ps.setFloat(15, (float) t1);
    ps.setFloat(16, (float) ct1);
    ps.setFloat(17, (float) ta1);

    ps.setFloat(18, (float) t2);
    ps.setFloat(19, (float) ct2);
```

```
ps.setFloat(20, (float) ta2);

ps.setFloat(21, (float) t3);
ps.setFloat(22, (float) ct3);
ps.setFloat(23, (float) ta3);
ps.setFloat(24, (float) t4);
ps.setFloat(25, (float) ct4);
ps.setFloat(26, (float) ta4);

ps.setFloat(27, (float) t5);
ps.setFloat(28, (float) ct5);
ps.setFloat(29, (float) ta5);

ps.setFloat(30, (float) internal1);
ps.setFloat(31, (float) external1);

ps.setFloat(32, (float) internal2);
ps.setFloat(33, (float) external2);

ps.setFloat(34, (float) internal3);
ps.setFloat(35, (float) external3);

ps.setFloat(36, (float) internal4);
ps.setFloat(37, (float) external4);

boolean rs = ps.execute();

if (rs == false) {
   out.println("<h2 align='center'>Resultl Processed Successfully </2>");
} else {
   out.println("<h2 align='center'>Error while processing result </2>");
}

connection.close();
```

```
jMenuItem3.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.
V K_A, java.awt.event.InputEvent.CTRL_MASK));
    jMenuItem3.setFont(new java.awt.Font("Segoe UI", 0, 16)); //
    NOI18N jMenuItem3.setText("Select All");
    jMenuItem3.addActionListener(new java.awt.event.ActionListener()
      { public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem3ActionPerformed(evt);
      }
    });
    jMenu2.add(jMenuItem3);


jMenuItem4.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.
V K_X, java.awt.event.InputEvent.CTRL_MASK));
    jMenuItem4.setFont(new java.awt.Font("Segoe UI", 0, 16)); // NOI18N
    jMenuItem4.setText("Cut");
    jMenuItem4.addActionListener(new java.awt.event.ActionListener()
      { public void actionPerformed(java.awt.event.ActionEvent evt) {
        jMenuItem4ActionPerformed(evt);
      }
    });
    jMenu2.add(jMenuItem4);


jMenuItem5.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.KeyEvent.
V K_C, java.awt.event.InputEvent.CTRL_MASK));
    jMenuItem5.setFont(new java.awt.Font("Segoe UI", 0, 16)); //
    NOI18N jMenuItem5.setText("Copy");
    jMenuItem5.addActionListener(new java.awt.event.ActionListener() {
```

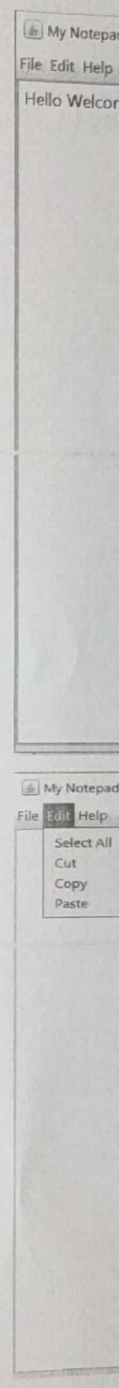
```
            a.awt.event.ActionEvent evt) {
        p       jMenuItem5ActionPerformed(evt);
    } u
} b
);
    l
    i jMenu2.add(jMenuItem5);
    c

jMenuItem6.setAccelerator(javax.swing.KeyStroke.getKeyStroke(java.awt.event.Key
Event.V K_V, java.awt.event.InputEvent.CTRL_MASK));
    ojMenuItem6.setFont(new java.awt.Font("Segoe UI", 0, 16)); //
    i NOI18N jMenuItem6.setText("Paste");
    djMenuItem6.addActionListener(new
        java.awt.event.ActionListener() { public void
    a   actionPerformed(java.awt.event.ActionEvent evt) {
    c       jMenuItem6ActionPerformed(evt);
} t
} i
);
    o
    n jMenu2.add(jMenuItem6);
    P
    e jMenuBar1.add(jMenu2);
    r
    f jMenu3.setText("Help");
    o jMenu3.setFont(new java.awt.Font("Segoe UI", 0, 16)); // NOI18N
    r
    m jMenuItem7.setFont(new java.awt.Font("Segoe UI", 0, 16)); // NOI18N
    e jMenuItem7.setText("About");
    d jMenuItem7.addActionListener(new
    (   java.awt.event.ActionListener() { public void
    j   actionPerformed(java.awt.event.ActionEvent evt) {
    a       jMenuItem7ActionPerformed(evt);
} v
});
```

```
        setJMenuBar(jMenuBar1);

jMenu3.add(jMenuItem7);
        javax.swing.GroupLayout layout = new
        javax.swing.GroupLayout(getContentPane()); getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
            ADING)
            .addComponent(jScrollPane1,
(javax.swing.GroupLayout.DEFAULT_SIZE, 1024, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
            ADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TR
AILING, layout.createSequentialGroup()
                .addComponent(jScrollPane1,
; javax.swing.GroupLayout.DEFAULT_SIZE, 678, Short.MAX_VALUE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code
        here: dispose();
    }

jMenu3.add(
        private void jMenuItem3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code
        here: jEditorPane1.selectAll();
);
```

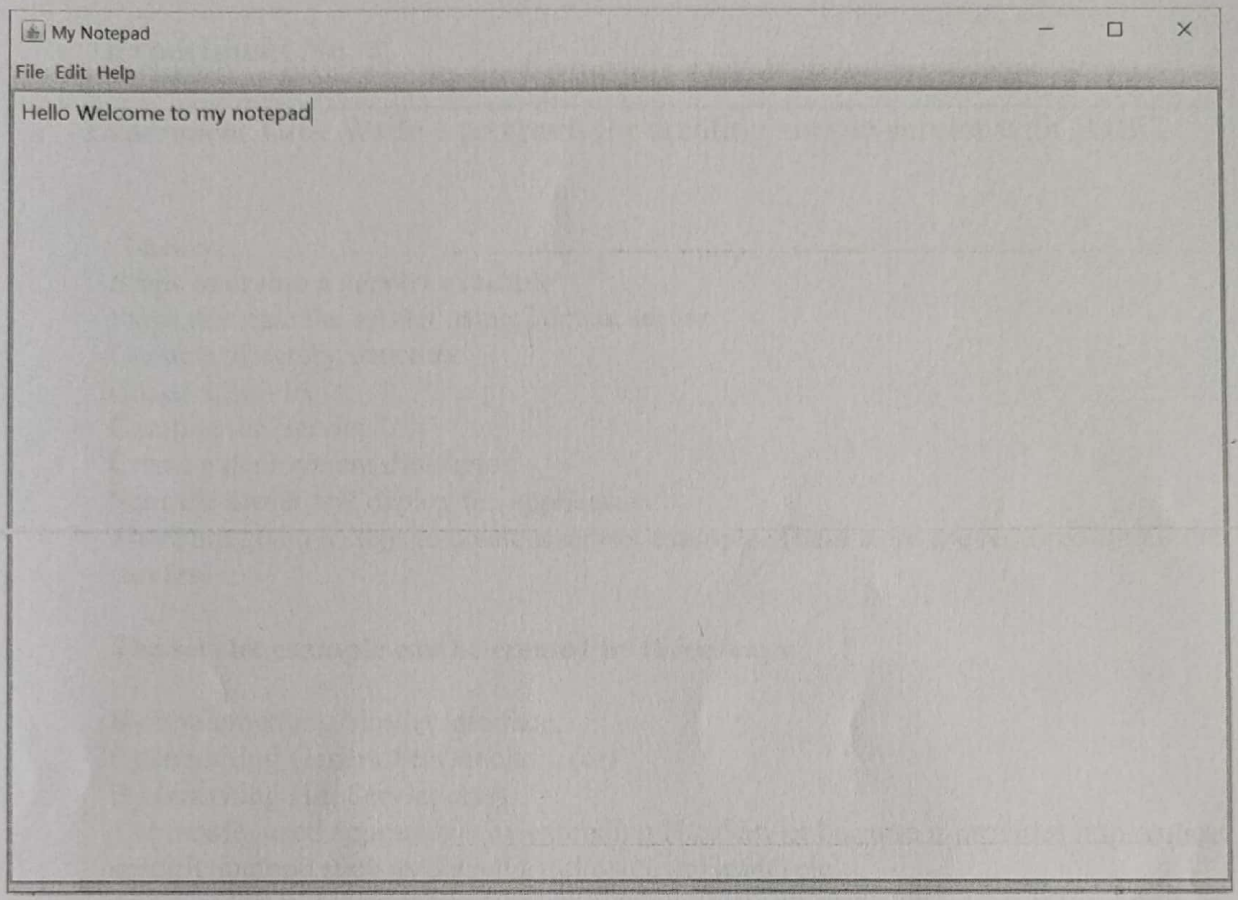
```
}        code here: jEditorPane1.copy();
    }

)
{    private void jMenuItem7ActionPerformed(java.awt.event.ActionEvent evt) {
/       // TODO add your handling code here:
/       jEditorPane1.setText("This is an simuation of notepad menu bar");
    }
T
O   private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
D       // TODO add your handling code
O       here: jEditorPane1.cut();
    }
a
d   private void jMenuItem6ActionPerformed(java.awt.event.ActionEvent evt) {
d       // TODO add your handling code
        here: jEditorPane1.paste();
y   }
o
u   /**
r    * @param args the command line arguments
     */
h    public static void main(String args[]) {
a
n       /* Set the Nimbus look and feel */
d   //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
l   /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and
i
n   * For details see
g   http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try
```

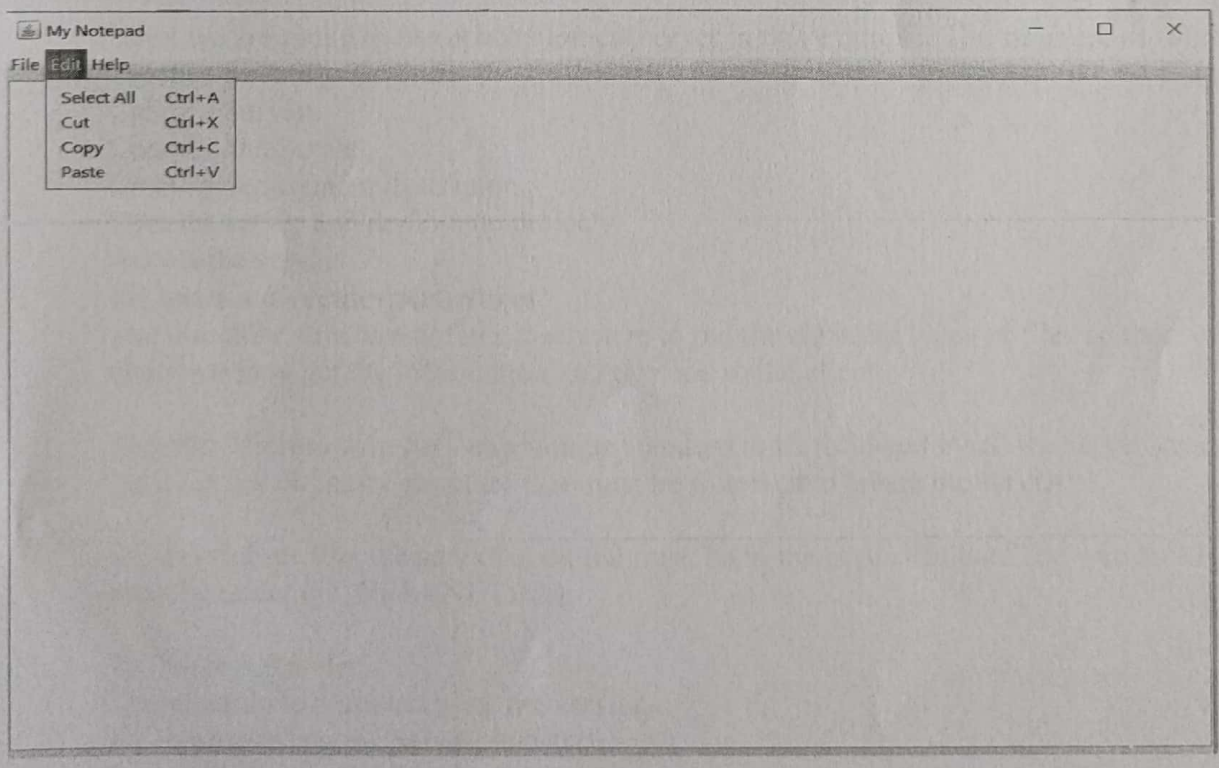
```
        {
            for (javax.swing.UIManager.LookAndFeelInfo
info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName()
                    ); break;
                }
            }
        } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Notepad.class.getName()).log(java.util.logging.Level.SE
VE RE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Notepad.class.getName()).log(java.util.logging.Level.SE
VE RE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Notepad.class.getName()).log(java.util.logging.Level.SE
VE RE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Notepad.class.getName()).log(java.util.logging.Level.SE
VE RE, null, ex);
        }
        //</editor-fold
```

```
    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable()
    {
        public void run() {
            new Notepad().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JEditorPane
jEditorPane1; private javax.swing.JMenu
jMenu1;
private javax.swing.JMenu jMenu2;
private javax.swing.JMenu jMenu3;
private javax.swing.JMenuBar
jMenuBar1;
private javax.swing.JMenuItem jMenuItem2;
private javax.swing.JMenuItem jMenuItem3;
private javax.swing.JMenuItem jMenuItem4;
private javax.swing.JMenuItem jMenuItem5;
private javax.swing.JMenuItem jMenuItem6;
private javax.swing.JMenuItem jMenuItem7;
private                javax.swing.JScrollPane
jScrollPane1;
// End of variables declaration
}
```

-: Output :-





Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 6

Experiment Title: Write a program for creating simple servlet with JDBC.

Theory:

Steps to create a servlet example

Steps to create the servlet using Tomcat server
Create a directory structure
Create a Servlet
Compile the Servlet
Create a deployment descriptor
Start the server and deploy the application
There are given 6 steps to create a servlet example. These steps are required for all the servers.

The servlet example can be created by three ways:

By implementing Servlet interface,
By inheriting GenericServlet class, (or)
By inheriting HttpServlet class
The mostly used approach is by extending HttpServlet because it provides http request specific method such as doGet(), doPost(), doHead() etc.

Here, we are going to use apache tomcat server in this example. The steps are as follows:
Create a directory structure
Create a Servlet
Compile the Servlet
Create a deployment descriptor
Start the server and deploy the project
Access the servlet

1)Create a directory structures

The directory structure defines that where to put the different types of files so that web container may get the information and respond to the client.

The Sun Microsystem defines a unique standard to be followed by all the server vendors. Let's see the directory structure that must be followed to create the servlet.

As you can see that the servlet class file must be in the classes folder. The web.xml file must be under the WEB-INF folder.

2)Create a Servlet

There are three ways to create the servlet.
By implementing the Servlet interface
By inheriting the GenericServlet class
By inheriting the HttpServlet class
The HttpServlet class is widely used to create the servlet because it provides methods to handle http requests such as doGet(), doPost, doHead() etc.
In this example we are going to create a servlet that extends the HttpServlet class. In this

example, we are inheriting the HttpServlet class and providing the implementation of the doGet() method. Notice that get request is the default request.

3)Compile the servlet

For compiling the Servlet, jar file is required to be loaded. Different Servers provide different jar files:

Jar file	Server
1) servlet-api.jar	Apache Tomcat
2) weblogic.jar	Weblogic
3) javaee.jar	Glassfish
4) javaee.jar	JBoss

Two ways to load the jar file
set classpath
paste the jar file in JRE/lib/ext folder
Put the java file in any folder. After compiling the java file, paste the class file of servlet in WEB-INF/classes directory.

4)Create the deployment descriptor (web.xml file)

The deployment descriptor is an xml file, from which Web Container gets the information about the servet to be invoked.

The web container uses the Parser to get the information from the web.xml file. There are many xml parsers such as SAX, DOM and Pull.

There are many elements in the web.xml file. Here is given some necessary elements to run the simple servlet program.
Description of the elements of web.xml file
There are too many elements in the web.xml file. Here is the illustration of some elements that is used in the above web.xml file. The elements are as follows:

<web-app> represents the whole application.
<servlet> is sub element of <web-app> and represents the servlet.
<servlet-name> is sub element of <servlet> represents the name of the servlet.
<servlet-class> is sub element of <servlet> represents the class of the servlet.
<servlet-mapping> is sub element of <web-app>. It is used to map the servlet.
<url-pattern> is sub element of <servlet-mapping>. This pattern is used at client side to invoke the servlet.

5)Start the Server and deploy the project

To start Apache Tomcat server, double click on the startup.bat file under apache-tomcat/bin directory.

Program :

Index.html

```
<html>
 <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Result Processing </title>
    <style>
      body{
        font-family: Calibri;
        background-color:
        #8EE4AF;
      }
      .container {
        padding: 50px;
        background-color:
        #5CDB95; padding-left:
        100px; padding-right:
        100px; margin-left:100px;
        margin-right: 100px;

      }

      input[type=text], input[type=password] {
        width: 98%;
        padding: 10px;
        margin: 5px 2px 5px 2px;
```

```
            display:
inline-block; border:
none; background:
#fff;


}
input[type=text]:focus, input[type=password]:focus {
  background-color: #AAEaA1;
  outline: none;
}
div {
  padding: 10px 0;
}
hr {
  border: 1px solid
  #f1f1f1; margin-bottom:
  25px;
}
.registerbtn {
  background-color: #05386B;
  color: white;
  padding: 16px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}
.registerbtn:hover {
  opacity: 1;
}
.maintable {
  background:
```

```
        #5CDB95;
      padding: 15px; margin:
        5px 0 5px 0;
        width: 100%;
      }
      select:focus {
        background-color:
        #AAEaA1; outline: none;
      }
      label {
        font-size: 18px;
        font-weight: bold;
      }
    </style>
    <script type="text/javascript">
      function validateform() {
        var id = document.register.id.value;
        if (isNaN(id)) {
          alert("Invaid
          ID"); return false;
        }
      }
    </script>
  </head>
  <body>
    <form name = "register" action="MyServlet" method="get" onsubmit="return
validateform()">
      <div class="container">
        <center> <h1 style="color:white;"> Result Processing</h1> </center>
        <hr>
          <table border="0" class="maintable" cellspacing="5" ><tr>

<td colspan="2" width="20%"><label> ID Number :</label></td>
            <td colspan="2"> <input type="text" name="id" placeholder= "ID
```

```
Number" size="15" required /></td>
            </tr>
            <tr>
              <td colspan="2"><label> Full name :</label></td>
              <td colspan="2"> <input type="text" name="fullname" placeholder=
"Full Name" size="15" required /> </td>
            </tr>
            <tr>
              <td colspan="2"><label> Course :</label> </td>
              <td colspan="2"><label> MCA </label> </td>
            </tr>
            <tr>
              <td><label> Class </label> </td>
              <td width="18%">: <input type="radio" name="class" value="FY"/> FY
                <input type="radio" name="class" value="SY" checked="checked"/> SY
                <input type="radio" name="class" value="SY"/> TY
              </td>
              <td width="10%"><label> Sem :</label></td>
              <td width="50%">
                <select name="Sem" >
                  <option value="I">I</option>
                  <option value="II">II</option>
                  <option value="III">III</option>
                  <option value="IV" selected>IV</option>
                  <option value="V">V</option>
                  <option value="VI">VI</option>
                </select>
              </td>
            </tr>
```

```
        } catch (Exception e) {
            out.println("<h2 align='center'>" + e + "</h2>");
        }
        out.println("</body>");
        out.println("</html>");
    }
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

@Override
public String getServletInfo()
    { return "Short description";
}// </editor-fold>

}
```

Database –> tblResult table schema:

```
create table tblResult
(
 id int primary key,
 name varchar(100),
 course varchar(10),
 class varchar(3),
 sem varchar(3),
 s1 varchar(40),
 s2 varchar(40),
 s3 varchar(40),
 s4 varchar(40),
 s5 varchar(40),
 s6 varchar(40),
 s7 varchar(40),
 s8 varchar(40),
 s9 varchar(40),
 t1 float,
ct1 float, ta1 float, t2 float, ct2
 float, ta2
  float, t3
 float, ct3
 float, ta3
  float, t4
 float, ct4
 float, ta4
  float, t5
 float, ct5
    float,
```

```
ta5 float,
i1 float,
e1 float,
i2 float,
c2 float,
i3 float,
e3 float,
i4 float,
e4 float
);
```

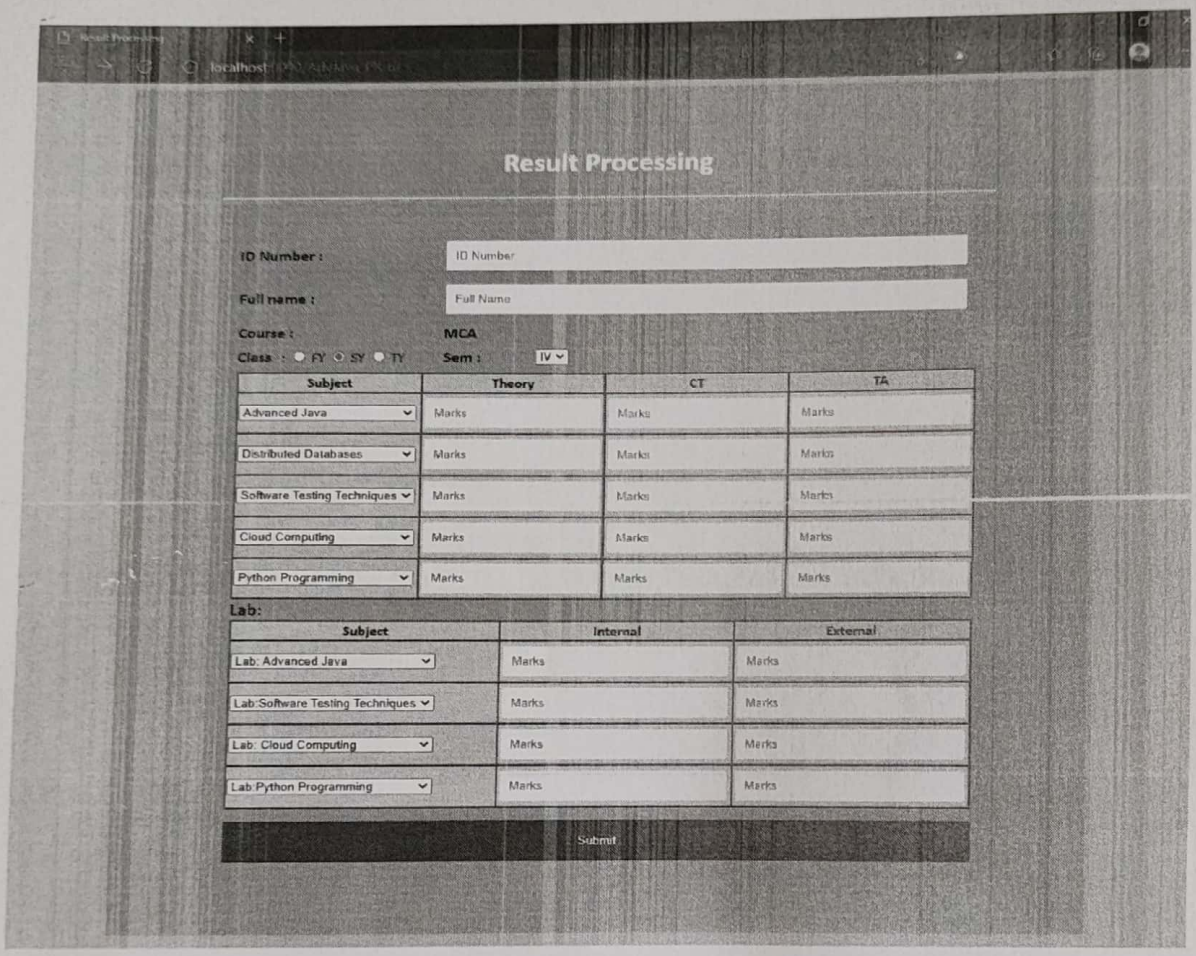
Database –> PL/SQL Stored Procedure:

```
CREATE OR REPLACE PROCEDURE
    inserttblResult( id IN tblResult.ID%TYPE,
    name IN tblResult.NAME%TYPE,
    course IN
    tblResult.COURSE%TYPE, class
    IN tblResult.CLASS%TYPE, sem
    IN tblResult.SEM%TYPE,

s1 IN tblResult.S1%TYPE, s2 IN tblResult.S2%TYPE, s3 IN tblResult.S3%TYPE, s4 IN
tblResult.S4%TYPE, s5 IN tblResult.S5%TYPE, s6 IN tblResult.S6%TYPE,


        s7                  IN
        tblResult.S7%TYPE,  s8
        IN  tblResult.S8%TYPE,
        s9                  IN
        tblResult.S9%TYPE,


    t1 IN tblResult.T1%TYPE,
        ct1 IN tblResult.CT1%TYPE,
```

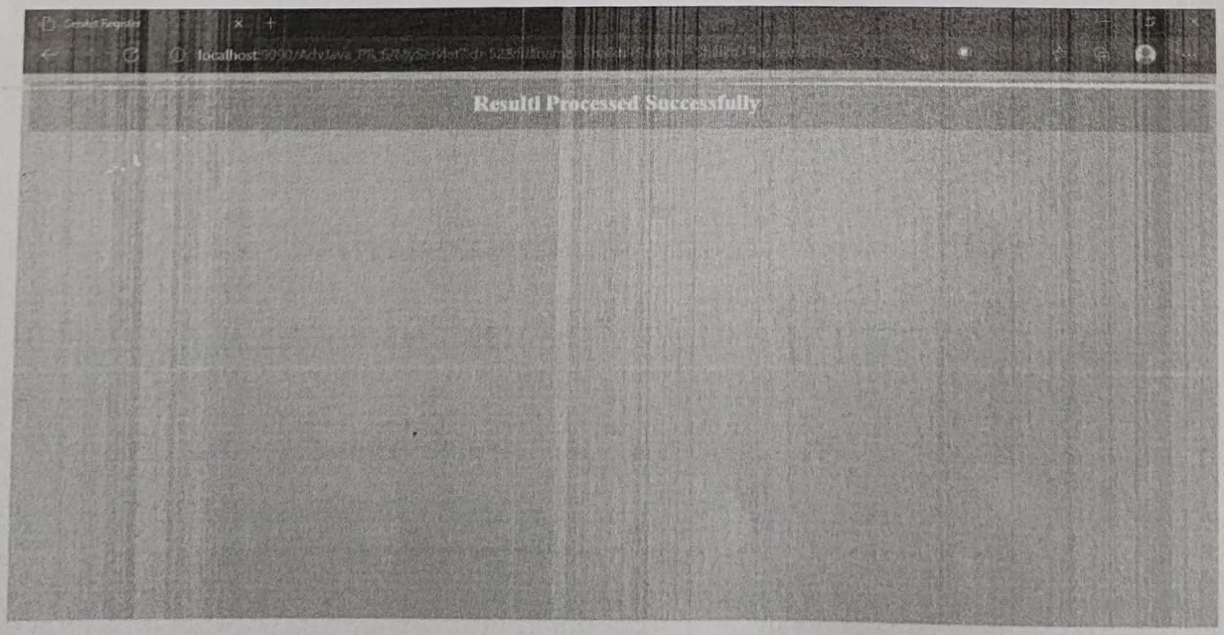
```
ta1 IN tblResult.TA1%TYPE,

t2 IN tblResult.T2%TYPE,
ct2 IN
tblResult.CT2%TYPE, ta2
IN tblResult.TA2%TYPE,

t3 IN tblResult.T3%TYPE,
ct3 IN
tblResult.CT3%TYPE, ta3
IN tblResult.TA3%TYPE,

t4 IN tblResult.T4%TYPE,
ct4 IN
tblResult.CT4%TYPE, ta4
IN tblResult.TA4%TYPE,

t5 IN tblResult.T5%TYPE,
ct5 IN
tblResult.CT5%TYPE, ta5
IN tblResult.TA5%TYPE,

i1 IN
tblResult.I1%TYPE, e1
IN tblResult.E1%TYPE,

        i2 IN
tblResult.I2%TYPE, e2
IN tblResult.E2%TYPE,

i3 IN
tblResult.I3%TYPE, e3
```

```
                IN tblResult.E3%TYPE,

                i4 IN
                tblResult.I4%TYPE, e4
                IN tblResult.E4%TYPE
                )
IS
BEGIN

 INSERT INTO
 tblResult VALUES
 (id,
name,course,class,sem,s1,s2,s3,s4,s5,s6,s7,s8,s9,t1,ct1,ta1,t2,ct2,ta2,t3,ct3,ta3,t4,ct4,ta4,t5
,ct5,ta5,i1,e1,i2,e2,i3,e3,i4,e4);

 COMMIT;

END;
/
```

-: Output



:-



Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 7

Experiment Title: Create Employee information Form using JSP.

Aim

Theory:

For creating registration form, you must have a table in the database. You can write the database logic in
JSP file, but separating it from the JSP page is better approach. Here, we are going to use DAO, Factory Method, DTO and Singletion design patterns. There are many files:

1.index.jsp for getting the values from the user
2.User.java, a bean class that have properties and setter and getter methods.
3.process.jsp, a jsp file that processes the request and calls the methods
4.Provider.java, an interface that contains many constants like DRIVER_CLASS, CONNECTION_URL, USERNAME and PASSWORD
5.ConnectionProvider.java, a class that returns an object of Connection. It uses the Singleton and factory method design pattern.
6.RegisterDao.java, a DAO class that is responsible to get access to the database

Program :

Index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Employee Information Form</title>
<script src="Script.js"></script>
<link rel="stylesheet" href="style.css" >
<link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600;800&displa
y=swa p" rel="stylesheet">
</head>
<body>
```

```
<img src="ss.PNG" />
<div class="container">
        <h1>Connect with Us </h1>
        <p>We would love to respond to your queries and help you succeed.Feel free
        to get in touch with us.</p>
        <div class="form-box">
        <div class="contact-left">
                <h3>Employee Information</h3>
                <form action="Registration" method="get" name="emp">
                <fieldset>
                <legend style="background-color: #ed4264;color: white; padding: 5px
                10px; margin-bottom: 7px;">Personal Information</legend>
                <div class="input-row">
               <div class="input-group">
                <label> Name</label>
                        <input type="text" placeholder="Name" name="name" required
                        />
                </div>

                <div class="input-group">
                        <label> Phone</label>
                        <input type="text" placeholder="Phone"
                        name="phone" onkeypress="return
                        isNumberKey(event)" required />
                </div>
                </div>

                <div class="input-row">
                <div class="input-group">
                        <label> Email</label>
                        <input type="email" name="email1" placeholder="Email"
                        required />
                </div>

                <div class="input-group">
```

```
                    <label> Aadhar No.</label>
                    <input type="text" placeholder="Aadhar No." name="aadhar"
                    onkeypress="return isNumberKey(event)"
                    onfocusout="return validateAadhar()" required
/>
    </div>
  </div>

  <div class="input-row">
    <div class="input-group">
      <label> Date of Birth</label>
      <input type="date" name="dob" required />
    </div>

    <div class="input-group">
      <label> Marital Status</label>
      <input type="text" name="ms" placeholder="Marital Status">
    </div>
  </div>

  <label> Address</label>
  <textarea rows="5" placeholder="Address" name="add" required></textarea>
  </fieldset>

  <fieldset>
    <legend style="background-color: #ed4264;color: white; padding: 5px 10px;
margin-bottom: 7px;">Job Information</legend>
    <div class="input-row">
    <div class="input-group">
      <label> Title</label>
      <input type="text" name="jt" placeholder="Job Title">
    </div>

    <div class="input-group">
```

```
        <label> Employee ID</label>
        <input type="text" placeholder="E ID" name="empID"
        onkeypress="return isNumberKey(event)" />
</div>
</div>

<div class="input-row">
<div class="input-group">
        <label> Supervisor</label>
        <input type="text" name="sv" placeholder="Supervisor">
</div>

<div class="input-group">
        <label> Department</label>
        <input type="text" name="dept" placeholder="Department">
</div>
</div>

<div class="input-row">
<div class="input-group">
        <label> Work Location</label>
        <input type="text" name="wl" placeholder="Work Location">
</div>

<div class="input-group">
        <label> Email</label>
        <input type="email" name="email2" placeholder="Email">
</div>
</div>

<div class="input-row">
<div class="input-group">
```

```
<label> Work Phone</label>

        <input type="text" name="wp"
        placeholder="Work Phone" onkeypress="return
        isNumberKey(event)" />

</div>


<div class="input-group">

        <label> Cell Phone</label>




        <input type="text" name="cp"
        placeholder="Cell Phone" onkeypress="return
        isNumberKey(event)" />

</div>

</div>


<div class="input-row">

<div class="input-group">

        <label> Start Date</label>

        <input type="date" name="startDate">

</div>


<div class="input-group">

        <label> Salary</label>

        <input type="text" name="sal" placeholder="Salary"
        onkeypress="return isNumberKey(event)" />

</div>

</div>

</fieldset>

<fieldset>

<legend style="background-color: #ed4264;color: white; padding: 5px
10px; margin-bottom: 7px;">Emergency Contact
Information</legend>

<div class="input-row">
```

```
                    <div class="input-group">
                        <label> Full Name</label>
                      <input type="text" name="rname" placeholder="Full name">
                    </div>

                    <div class="input-group">
                      <label> Relation</label>
                      <input type="text" name="rel" placeholder="Relation">
                    </div>
                  </div>



                  <div class="input-row">
                    <div class="input-group">
                      <label>Primary Phone</label>
                      <input type="text" name="pp" placeholder="Primary phone"
onkeypress="return isNumberKey(event)"/>
                    </div>

                    <div class="input-group">
                      <label> Alternate Phone</label>
                      <input type="text" name="ap" placeholder="Alternate phone"
onkeypress="return isNumberKey(event)" />
                    </div>
                  </div>
                  <label> Address</label>
                  <textarea rows="5" name="raddress" placeholder="Address"></textarea>
                  </fieldset>
                  <button type="submit" >Register</button>
                </form>
              </div>
            </div>
```

```
    </div>
  </body>
</html>
```

Script.js

```
function isNumberKey(evt)
{
  var charCode = (evt.which) ? evt.which :
  event.keyCode; if (charCode != 46 && charCode > 31
       && (charCode < 48 || charCode > 57))
    return false;
  return true;
}

function validateAadhar()
{
  ad = document.emp.aadhar.value;
  if (ad.length != 12){
    alert("Invalid Aadhar
    No.") return false;
  }
  return true;
}
```

Style.css

```
*{
    margin: 0;
    padding: 0;
}

body{
    background-image: linear-gradient(to top, #ed4264, #ffedbc);
    font-size: 16px;
    font-family: 'Poppins', sans-serif;
}

.container{
    width:
    80%;
    margin: 50px auto;
}

.form-box{
    background: #fff;
    display: flex;
}

.contact-left{
    flex-basis: 100%;
    padding: 40px 60px;
}

h1{
    margin-bottom: 10px;
    border: 3px solid white;
```

```
    width: 38vh;
    border-radius: 100vh;
    padding-left: 10px;
    color: #ED4264;
}

.container p{
    margin-bottom: 40px;
}

.input-row{
    display:
    flex;
    justify-content: space-between;
    margin-bottom: 20px;

}

.input-row .input-group{
    flex-basis: 45%;

}
input{
    width: 100%;
    border: none;
    border-bottom: 1px solid #ccc;
    outline: none;
    font-size: 14px;
    padding-bottom: 5px;
```

```
}

textArea{
    width: 100%;
    border: 1px solid #ccc;
    outline: none;
    padding:10px;
    box-sizing: border-
    box; resize: none;
}

label{
    margin-bottom: 6px;
    display: block;
    color: #ed4264;
}

button{
    background: #ed4264;
    width: 150px;
    border: none;
    outline: none;
    color: #fff;
    height: 35px;
    border-radius: 30px;
    margin-top: 20px;
    box-shadow: 0px 5px 15px 0px
    rgba(28,0,181,0.3); cursor: pointer;
    left: 50%;
    position: relative;
    -ms-transform: translate(-50%);
    transform: translate(-50%); }
```

```
.contact-left h3{
color: #ed4264;
font-weight: 600;
margin-bottom: 30px;
 }


 tr td:first-child{
    padding-right: 20px;
 }

 tr td{
    padding-top: 20px;
 }

 fieldset{
    padding: 10px;
    margin-top: 10px;
 }

 img{
    float: right;
    width:100px;
    height:100px;
    margin-right:
    20px; margin-top:
    0px;
    border-radius: 100px;
    opacity: 0.8;
    box-shadow: 0px 0px 10px 8px #ed4264;
 }
```

DatabaseConn.java

```
package backEnd;

import java.sql.*;

public class DatabaseConn {

   public String dbConn(String
        name, long phone,
        String personal_email,
        long aadhar,
        String dob,
        String marital_status,
        String address,
        -[=00String
        job_title, int
        eid,
        String
        supervisor,
        String dept,
        String work_loc,
        String office_email,
        long work_phone,
        long cell_phone,
        String job_start_date,
        float salary,
        String relative_name,
        String relation,
long relative_phone, long
```

```
        relative_alt_phone,
        String relative_address)
        {

      String msg =
      null; try {
        Connection con = null;
        Class.forName("com.mysql.cj.jdbc.Driver"
        ); con = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/dbSameerOrganization", "sameer", "root");

        PreparedStatement ps = con.prepareStatement(
            "call InsertEmp(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?);");

ps.setString(1, name); ps.setLong(2, phone); ps.setString(3, personal_email); ps.setLong(4, aadhar);
ps.setString(5, dob); ps.setString(6, marital_status); ps.setString(7, address); ps.setString(8, job_title);
ps.setInt(9, eid); ps.setString(10, supervisor); ps.setString(11, dept); ps.setString(12, work_loc);
ps.setString(13, office_email); ps.setLong(14, work_phone); ps.setLong(15, cell_phone);
ps.setString(16, job_start_date); ps.setFloat(17, salary); ps.setString(18, relative_name); ps.setString(19,
relation); ps.setLong(20,
relative_phone);
ps.setLong(21, relative_alt_phone);

        ps.setString(22, relative_address);

        int i = ps.executeUpdate();
        con.close();
        if (i > 0) {
          msg = "You are successfully registered...";
        } else {
          msg = "Something went wrong...! Please try again or contact admin";
        }

      } catch (ClassNotFoundException | SQLException e)
```

```
        { System.out.println("DBException : "+e);
      }
      return msg;
    }

}
```

Registration.java (Servlet)

```
package backEnd;

import
java.io.IOException;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Registration extends HttpServlet {

   protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Register</title>");
```

```
        out.println("</head>");
        out.println("<body>");

String name, personal_email, marital_status, address, job_title,

        supervisor, dept, work_loc, office_email, relative_name, relation,
relative_address;
        long phone, aadhar, work_phone, cell_phone, relative_phone, relative_alt_phone;

        int eid;
        float salary;
        String dob, job_start_date;
        String temp;

        try {
          name = request.getParameter("name");
          phone = Long.parseLong(request.getParameter("phone"));
          personal_email = request.getParameter("email1");
          aadhar = Long.parseLong(request.getParameter("aadhar")); dob
          = request.getParameter("dob");
          marital_status = request.getParameter("ms");
          address = request.getParameter("add");

          job_title = request.getParameter("jt");

          if ((temp = request.getParameter("empID")).equals("")) {
            eid = Integer.parseInt("0");
          } else {
            eid = Integer.parseInt(temp);
          }

          supervisor = request.getParameter("sv");
          dept = request.getParameter("dept");
          work_loc = request.getParameter("wl");
          office_email = request.getParameter("email2");
```

```
if ((temp = request.getParameter("wp")).equals("")) {
   work_phone = Long.parseLong("0");
} else {
   work_phone = Long.parseLong(temp);
                        }

if ((temp = request.getParameter("cp")).equals("")) {
   cell_phone = Long.parseLong("0");
} else {
   cell_phone = Long.parseLong(temp);
}
job_start_date = request.getParameter("startDate");

if ((temp = request.getParameter("sal")).equals("")) {
   salary = Float.parseFloat("0");
} else {
   salary = Float.parseFloat(temp);
}

relative_name = request.getParameter("rname");
relation = request.getParameter("rel");

if ((temp = request.getParameter("pp")).equals("")) {
   relative_phone = Long.parseLong("0");
} else {
   relative_phone = Long.parseLong(temp);
}
if ((temp = request.getParameter("ap")).equals("")) {
   relative_alt_phone = Long.parseLong("0");
} else {
   relative_alt_phone = Long.parseLong(temp);
```

```
        }

        relative_address = request.getParameter("raddress");

        DatabaseConn dbc = new DatabaseConn();
        String msg = dbc.dbConn(name, phone, personal_email, aadhar, dob,
marital_status, address, job_title, eid, supervisor, dept,
            work_loc, office_email, work_phone, cell_phone, job_start_date, salary,
relative_name, relation, relative_phone, relative_alt_phone, relative_address);

        out.println("<style>");
        out.println("body{");
        out.println("background-image: linear-gradient(to top, #ed4264, #ffedbc);");
        out.println("height: 100vh;");
        out.println("}");
        out.println("h1{");
        out.println("background-color: #ed4264;");
        out.println("padding: 15px;");
        out.println("margin: 15px 20px 22px
        20px;"); out.println("color:white;");
        out.println("}");
        out.println("</style>");

        if (msg == null) {
          out.println("<h1 align='center'>Already registered.</h1>");
        } else {
          out.println("<h1 align='center'>" + msg + "</h1>");
        }

        out.println("</body>");
        out.println("</html>");

      } catch (Exception e) {
        PrintWriter pw = response.getWriter();
        pw.println("Exception : " + e);
```

```
        }

    }
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

@Override
public String getServletInfo()
    { return "Short description";
}// </editor-fold>
```

Database (Schemas, procedure):

```
CREATE DATABASE dbSameerOrganization;

CREATE TABLE
tblDept (
        dept VARCHAR(30),
        supervisor
        VARCHAR(40),
        work_loc
        VARCHAR(20)
);

CREATE TABLE
tblEmp (
        aadhar BIGINT,
        name
        VARCHAR(40),
        job_title VARCHAR(30),
        personal_email
        VARCHAR(40), office_email
        VARCHAR(40), phone
        BIGINT,
        work_phone
        BIGINT, cell_phone
        BIGINT, dob
        DATE,
        marital_status
        VARCHAR(10), address
        VARCHAR(60),
eid INT, job_start_date DATE, salary
```

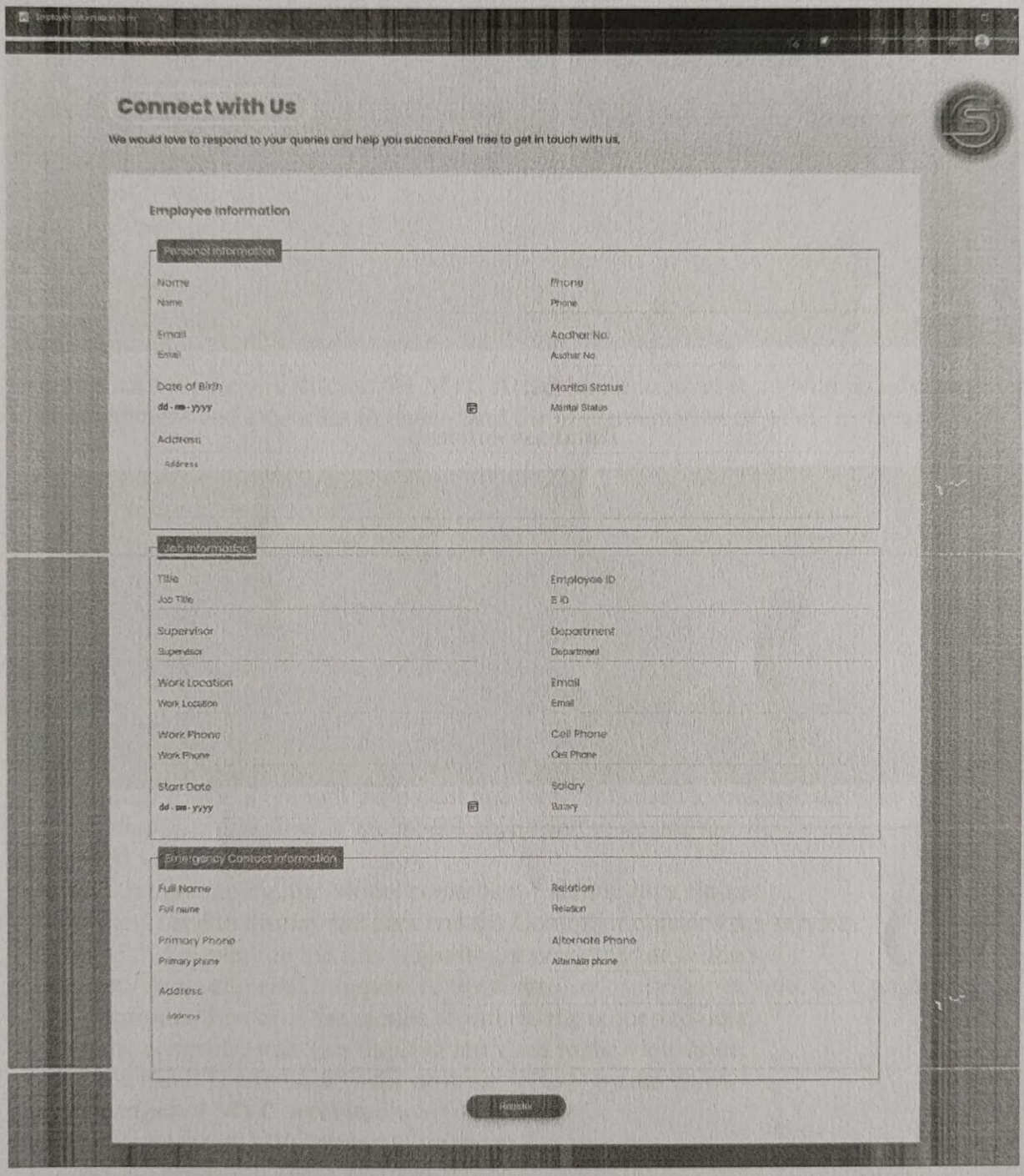
```
        FLOAT,
        dept VARCHAR(30),
        PRIMARY KEY(aadhar)
);

CREATE TABLE
tblEmpRelative (
        relative_name VARCHAR(40),
        relation VARCHAR(30),
        relative_phone BIGINT,
        relative_alt_phone BIGINT,
        relative_address
        VARCHAR(60), emp_aadhar
        BIGINT,

        FOREIGN KEY(emp_aadhar) REFERENCES tblEmp(aadhar)
);

DELIMITER &&
create procedure InsertEmp(
        IN name
        VARCHAR(40),
        IN phone BIGINT,
        IN personal_email
        VARCHAR(40), IN aadhar
        BIGINT,
        IN dob VARCHAR(12),
        IN marital_status
        VARCHAR(10), IN address
        VARCHAR(60),

        IN job_title
        VARCHAR(30), IN eid
```

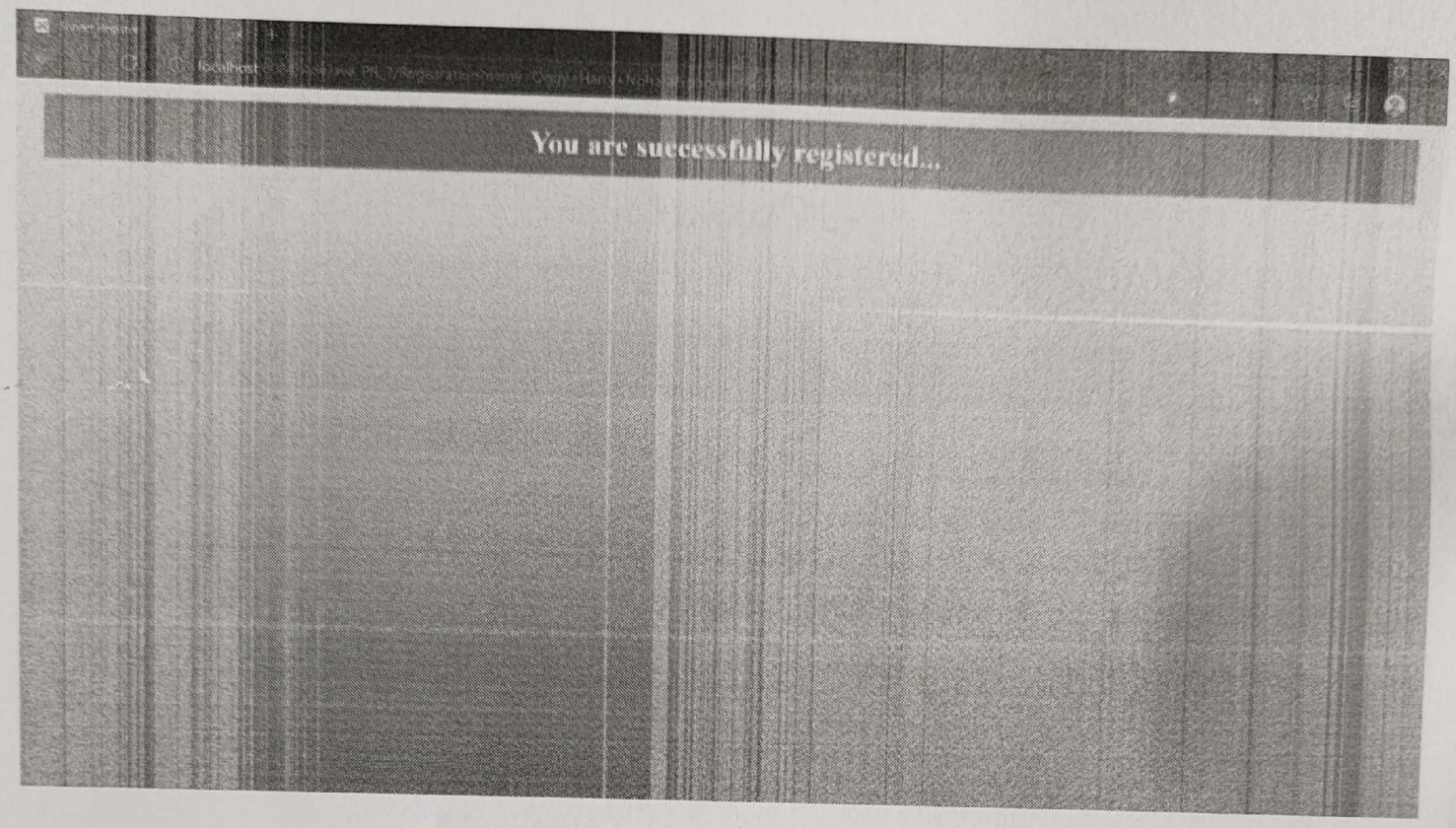
```
        INT,
        IN supervisor
    VARCHAR(40), IN dept
    VARCHAR(30),
        IN work_loc VARCHAR(20),
        IN office_email


        VARCHAR(40), IN
        work_phone BIGINT,
        IN cell_phone BIGINT,
        IN job_start_date VARCHAR(12),
        IN salary FLOAT,


        IN relative_name
        VARCHAR(40), IN relation
        VARCHAR(30),
        IN relative_phone BIGINT,
        IN relative_alt_phone BIGINT,
        IN relative_address VARCHAR(60))

    BEGIN
     INSERT INTO tblDept
     VALUES(dept,supervisor,work_loc
     );


          INSERT INTO tblEmp
          VALUES (aadhar, name, job_title ,personal_email, office_email, phone ,work_phone
    ,cell_phone, STR_TO_DATE(dob, '%Y-%m-%d'), marital_status, address,
    eid, STR_TO_DATE(job_start_date, '%Y-%m-%d'), salary, dept);


          INSERT INTO tblEmpRelative
          VALUES (relative_name, relation, relative_phone,
    relative_alt_phone, relative_address, aadhar);
          END
```

&&

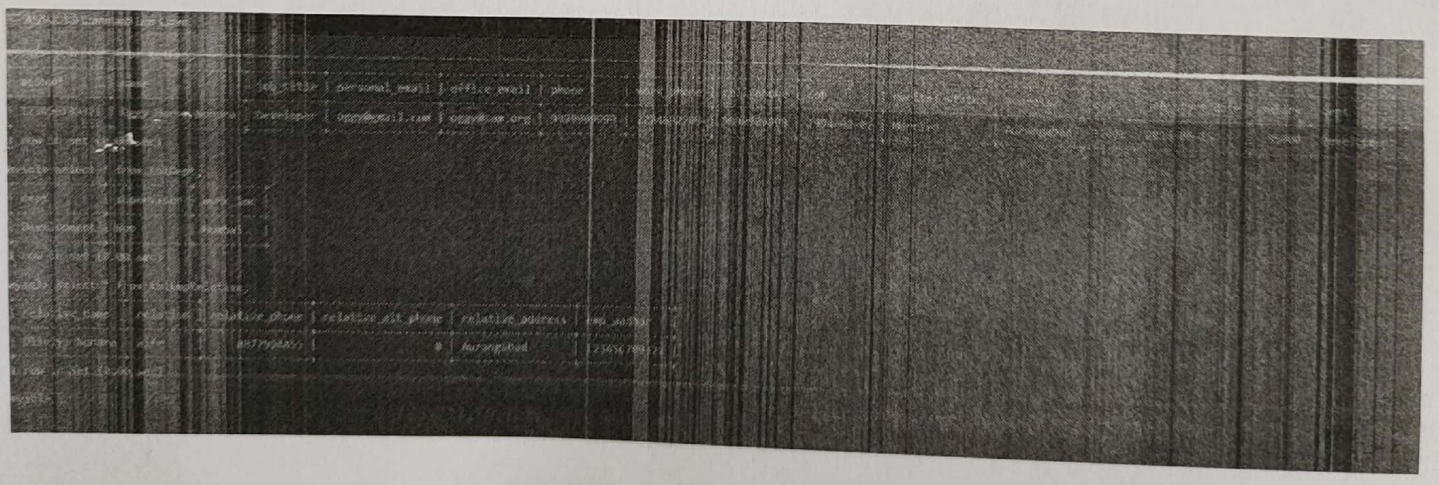
DELIMITER ;

-: Output :-





-: Database entries :-



Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 8

Aim: Write a program for implementing concept of MVC Architecture.

Theory:

MVC Architecture in Java

The Model-View-Controller (MVC) is a well-known design pattern in the web development field. It is way to organize our code. It specifies that a program or application shall consist of data model, presentation information and control information. The MVC pattern needs all these components to be separated as different objects.

In this section, we will discuss the MVC Architecture in Java, alongwith its advantages and disadvantages and examples to understand the implementation of MVC in Java.

What is MVC architecture in Java?

The model designs based on the MVC architecture follow MVC design pattern. The application logic is separated from the user interface while designing the software using model designs.

The MVC pattern architecture consists of three layers:

Model: It represents the business layer of application. It is an object to carry the data that can also contain the logic to update controller if data is changed.

View: It represents the presentation layer of application. It is used to visualize the data that the model contains.

Controller: It works on both the model and view. It is used to manage the flow of application, i.e. data flow in the model object and to update the view whenever data is changed.

In Java Programming, the Model contains the simple Java classes , the View used to display the data and the Controller contains the servlets . Due to this separation the user requests are processed as follows:

A client (browser) sends a request to the controller on the server side, for a page.

The controller then calls the model. It gathers the requested data.

Then the controller transfers the data retrieved to the view layer.

Now the result is sent back to the browser (client) by the view.

Advantages of MVC Architecture

The advantages of MVC architecture are as follows:

MVC has the feature of scalability that in turn helps the growth of application.

The components are easy to maintain because there is less dependency.

A model can be reused by multiple views that provides reusability of code.

The developers can work with the three layers (Model, View, and Controller) simultaneously.

Using MVC, the application becomes more understandable.

Using MVC, each layer is maintained separately therefore we do not require to deal with massive code.

The extending and testing of application is easier.
Implementation of MVC using Java
To implement MVC pattern in Java, we are required to create the following three classes.

Employee Class, will act as model layer
EmployeeView Class, will act as a view layer
EmployeeContoller Class, will act a controller layer

MVC Architecture Layers

Model Layer

The Model in the MVC design pattern acts as a data layer for the application. It represents the business logic for application and also the state of application. The model object fetch and store the model state in the database. Using the model layer, rules are applied to the data that represents the concepts of application.

View Layer

As the name depicts, view represents the visualization of data received from the model. The view layer consists of output of application or user interface. It sends the requested data to the client, that is fetched from model layer by controller.

Controller Layer

The controller layer gets the user requests from the view layer and processes them, with the necessary validations. It acts as an interface between Model and View. The requests are then sent to model for data processing. Once they are processed, the data is sent back to the controller and then displayed on the view.

Main Class Java file
the main file to implement the MVC architecture. Here, we are using the MVCMain class.

Program :

```
<html>
<head>
<title>Login Form</title>
<link rel="stylesheet" type="text/css" href="Style.css">
</head>
<body>
<h2>Login Page</h2><br>
<div class="login">
<form method="post" action="LoginController">
<label><b>User Name
</b>
</label>
<input type="text" name="Uname" id="Uname" placeholder="Username">
<br><br>
<label><b>Password </b>
```

```
</label>
<input type="Password" name="Pass" id="Pass" placeholder="Password">
<br><br>
<input type="submit" name="log" id="log" value="Log In Here">
</form>
</div>
</body>
```

Style.css

```
body
{
background-image: linear-gradient(to top, #4CF550,
whitesmoke); font-family: 'Arial';
height: 100vh;
}
.login{
width: 22vw;
height: 22vh;
overflow:
hidden;
margin: auto;
padding:
80px;
background:
#23663C; border-
radius: 15px ;
}
h2{
  text-align:
  center;
  color:#23663C;
```

```
    padding: 20px;
    margin-bottom:
    0; margin-top:
    8vh; font-size:
    3vw;
}
label{
    color: white;
    fontsize:15px;
}
#Uname{
    width:
    20vw;
    height: 5vh;
    border: none;
    border-radius:
    3px; padding-
    left: 8px;
}
#Pass{
    width: 20vw;
    height: 5vh;
    border: none;
    border-radius:
    3px; padding-
    left: 8px;
}
#log{
    width:
    20vw;
    height: 5vh;
```

```
  border:
  none;
  border-radius: 17px;
  padding-left: 10px;
  color: white;
  background:
  #4CF550; font-
  weight: bolder;
}
span{
  color: white;
  font-size:
  17px;
}
```

login-success.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Login Successful</title>
    <link rel="stylesheet" type="text/css" href="Style.css">

    <script
      type="text/javascript">
      function noBack()
      {
        window.history.forward()
      }
```

```
        noBack();
        window.onload = noBack;
        window.onpageshow = function
        (evt) {
          if
            (evt.persiste
            d) noBack();
        };
        window.onunload = function
          () { void (0);
        };
    </script>
  </head>
  <body>
    <%@page import="main.LoginBean"%>

    <h2>
            <%
        LoginBean bean = (LoginBean)
        request.getAttribute("bean"); out.print("Welcome, " +
        bean.getName()); bean.setName(null);
        bean.setPassword(null);
      %>
    </h2>

    <a href="index.html" style="text-decoration:none;">
      <button id="log" style="margin:auto; display:block;margin-top: 20px;
background: red;" onclick="noBack()">Log out</button>
    </a>
  </body>
</html>
```

login-error.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Login Error</title>
  </head>
  <body>
    <h2 style="color:red">Sorry! username or password error</h2>
    <%@ include file="index.html" %>
  </body>
</html>
```

LoginBean.java

```
package main;
import java.io.Serializable;

public class LoginBean implements Serializable {

  private String name, password;

  public String
    getName() { return
    name;
  }

  public void setName(String
    name) { this.name = name;
  }
```

```
    public String
      getPassword() { return
      password;
    }

    public void setPassword(String
      password) { this.password =
      password;
    }
}
```

LoginController.java

```
package main;

import
java.io.IOException;
import
java.io.PrintWriter;
import
javax.servlet.RequestDispatcher;
import
javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;

public class LoginController extends HttpServlet {
```

```
protected void processRequest(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
  response.setContentType("text/html;charset=UTF-
  8"); try (PrintWriter out = response.getWriter()) {
    /* TODO output your page here. You may use following sample
    code. */ String name = request.getParameter("Uname");
    String password = request.getParameter("Pass");

    LoginBean bean = new
    LoginBean();
    bean.setName(name);
    bean.setPassword(password);
    request.setAttribute("bean", bean);
    LoginDAO LD = new
    LoginDAO(bean); boolean status
    = LD.validateLogin();

    if (status) {
      RequestDispatcher rd = request.getRequestDispatcher("login-success.jsp");
                          rd.forward(request, response);
    } else {
      RequestDispatcher rd = request.getRequestDispatcher("login-
      error.jsp"); rd.forward(request, response);
    }

  }
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
    response) throws ServletException, IOException {
```

```
        processRequest(request, response);
    }

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        processRequest(request, response);
    }

    @Override
    public String
        getServletInfo() { return
        "Short description";
    }
}
```

LoginDAO.java

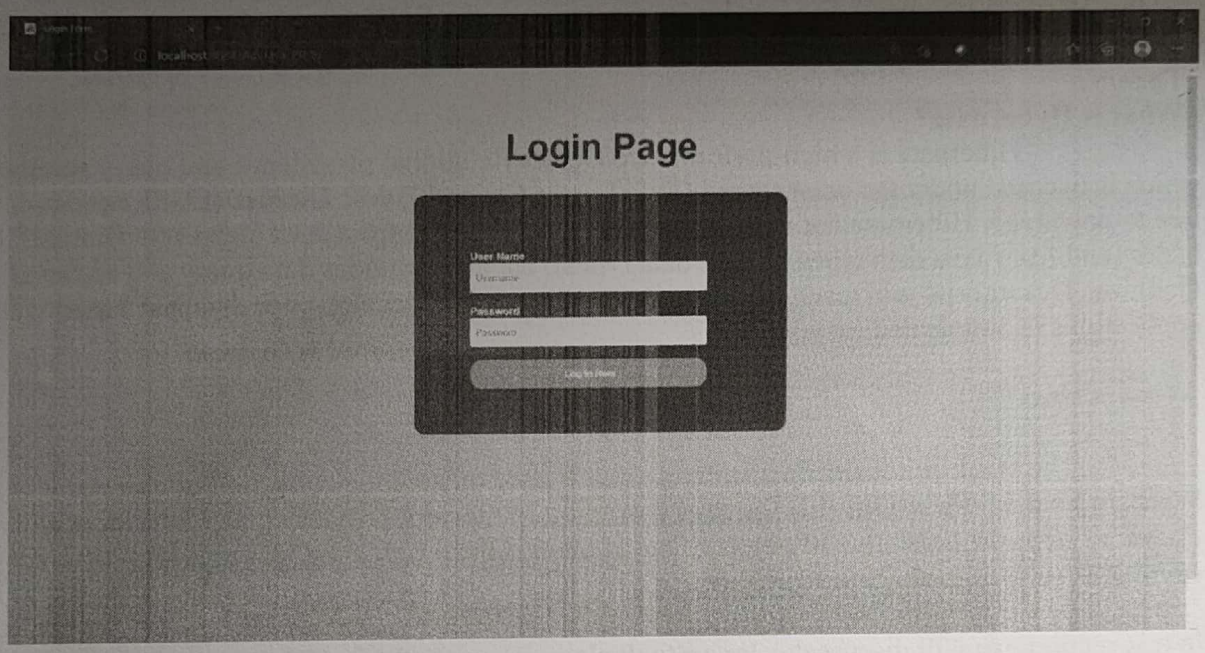
```
package main;
import
java.sql.*;

public class LoginDAO {

  String user,
  passwd;
  String u, p;
  boolean flag =
  false; ResultSet rs;

  LoginDAO(LoginBean bean)
    { user = bean.getName();
```

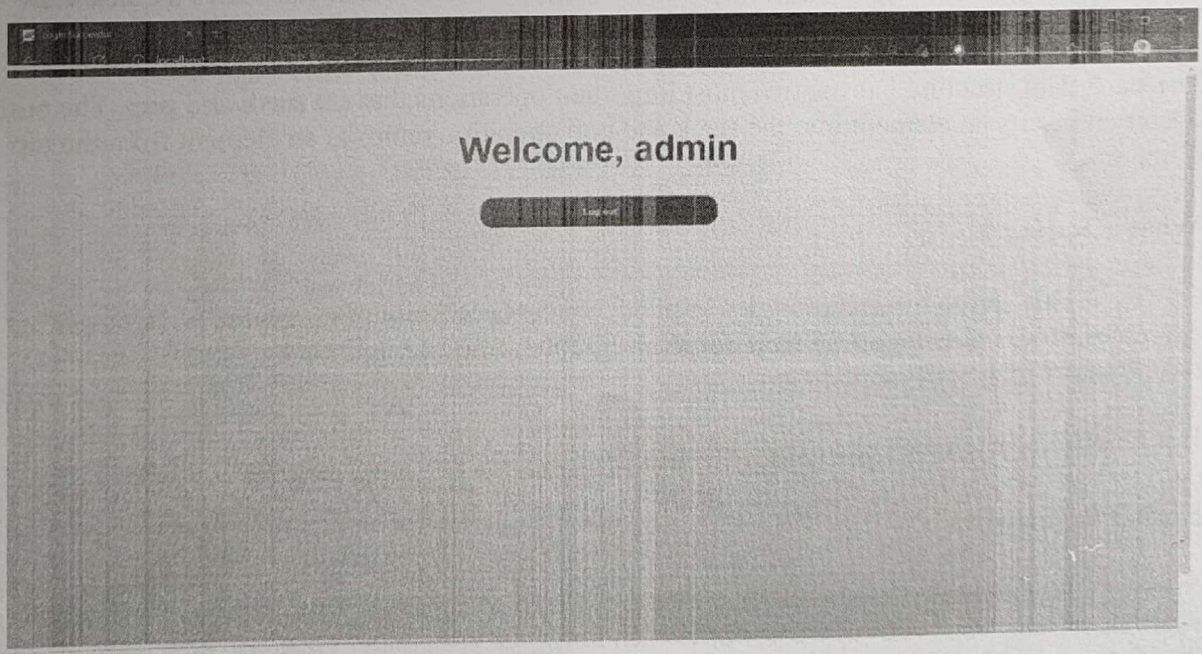
```
    passwd =
    bean.getPassword();
  }

  public boolean
    validateLogin() { try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      Connection con =
      DriverManager.getConnection(
          "jdbc:mysql://localhost:3306/login", "sameer", "root");

      Statement stmt = con.createStatement();
      rs = stmt.executeQuery("SELECT * FROM tblLogin WHERE username Like '"
      + user + "'"); while (rs.next()) {
        u =
        rs.getString(1); p
        = rs.getString(2);
      }
      con.close();

      if (user.equalsIgnoreCase(u) && passwd.equals(p)) {
            return true;
      }

    } catch (ClassNotFoundException |
      SQLException e) { System.out.println(e);
    }
    return false;
  }
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.1" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web- app_3_1.xsd">
  <servlet>
    <servlet-name>LoginController</servlet-name>
    <servlet-class>main.LoginController</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LoginController</servlet-name>
    <url-pattern>/LoginController</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-
      timeout> 30
    </session-timeout>
  </session-config>
</web-app>
```

-: Output :-





Class: SY (MCA) **Subject: Lab I- Advance Java**

Experiment No. 9

Aim: Write a program for implementing concept of Hibernate, Stuct, and spring.

Theory:

What is Hybernate?

Hibernate is a high-performance Object/Relational persistence and query service, which is licensed under the open source GNU Lesser General Public License (LGPL) and is free to download. Hibernate not only takes care of the mapping from Java classes to database tables (and from Java data types to SQL data types), but also provides data query and retrieval facilities. This tutorial will teach you how to use Hibernate to develop your database based web applications in simple and easy steps.

What is stuct?

The **stack** is a linear data structure that is used to store the collection of objects. It is based on **Last-In-First-Out** (LIFO). Java collection framework provides many interfaces and classes to store the collection of objects. One of them is the **Stack class** that provides different operations such as push, pop, search, etc.

In this section, we will discuss the **Java Stack class**, its **methods,** and **implement** the stack data structure in a Java program. But before moving to the Java Stack class have a quick view of how the stack works.

The stack data structure has the two most important operations that are **push** and **pop**. The push operation inserts an element into the stack and pop operation removes an element from the top of the stack.

What is Spring ?

This spring tutorial provides in-depth concepts of Spring Framework with simplified examples. It was **developed by Rod Johnson in 2003**. Spring framework makes the easy development of JavaEE application.

It is helpful for beginners and experienced persons.

Spring Framework

Spring is a *lightweight* framework. It can be thought of as a *framework of frameworks* because it provides support to various frameworks such as Struts, Hibernate, Tapestry, EJB, JSF, etc. The framework, in broader sense, can be defined as a structure where we find solution of the various technical problems.

The Spring framework comprises several modules such as IOC, AOP, DAO, Context, ORM, WEB MVC etc. We will learn these modules in next page. Let's understand the IOC and Dependency Injection first.

Program :

StudentRegistration.jsp

```
<%@page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Student Registration</title>
    <style>
        body{
            font-family: Calibri, Helvetica, sans-serif;
            background-color: white;
            }
        .container
        {
            font-family: Calibri, Helvetica, Times New Roman;
            font-size: 18px;
            padding: 40px;
            background-color: #AFD275;
            width: 45vw;
            align: center;
            margin-top: 20px;
            margin-left: 25%;
        }
        label{
            align-items: center;
            color: #1F2833;
        }
        input[type=text], input[type=password], textarea
        {
            width: 95%;
            padding: 15px;
            margin: 5px 0 20px 0;
            display: inline-block;
            border: none;
            background: #f1f1f1;
            align-items: center;
        }
        input[type=text]:focus, input[type=password]:focus
        {
```

```
        }
        div
        {
            padding: 10px 0;
        }
        hr
        {
            border: 1px solid #E7717D;
            margin-bottom: 25px;
        }
        .Register
        {
            background-color: #1F2833;
            color: white;
            padding: 16px 20px;
            margin-left: 250px;
            margin-right: 200px;
            border: none; cursor:
            pointer; width: 20%;
            opacity: 0.9;
            border-radius: 12px;
            font-weight: bold;
        }
        .Register:hover
        {
            opacity: 3;
        }
    </style>
</head>
<body>
    <form action="addStudent">
        <div class="container">
            <center><h1>Student Registration Form</h1></center>
            <label><b>Roll No :</b></label> <br>
                <input type="text" name="sid" required/><br />
            <label><b>Student Name :</b></label><br>
                <input type="text" name="sname" /><br />
            <label><b>Class :</b></label><br>
                <input type="text" name="sclass" required /><br />
            <label><b>Username :</b></label> <br>
                <input type="text" name="suname" required /><br />
            <label><b>Password :</b></label> <br>
                <input type="password" name="spass" required /><br />
            <label><b>Subject :</b></label> <br>
                <input type="text" name="ssub" required /><br />
            <input type="submit" value="Register" class="Register" />
        </div>
    </form>

</body>
</html>
```

studentRegistrationResponse.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1" isELIgnored="false" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Response</title>
    <style>
    h2 {
        background-color:
        #AFD275; padding: 15px;
        margin: 5px 0 22px
        0; color: #1F2833;
        text-align: center;
    }
    </style>
</head>
<body>
    <h2>${msg}</h2>
</body>
</html>
```

MyController.java

```
package com.sameer.HibernateSpringMVC;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam; import
org.springframework.web.servlet.ModelAndView;

import com.sameer.HibernateSpringMVC.student.AddStudent; import
com.sameer.HibernateSpringMVC.student.entities.Student; import
com.sameer.HibernateSpringMVC.teacher.entities.Subject;

@Controller
public class MyController {

@RequestMapping("addStudent")
    public ModelAndView addStudent(HttpServletRequest request,
HttpServletResponse response)
```

```
{
        ModelAndView mv = new ModelAndView();
        Student student = new Student(); Subject
        subject = new                    Subject();
        AddStudent addStudent = new AddStudent();

        String sId, sName, sClass, sUsername, sPassword; sId =
        request.getParameter("sid");
        sName = request.getParameter("sname"); sClass =
        request.getParameter("sclass"); sUsername =
        request.getParameter("suname"); sPassword =
        request.getParameter("spass");
        subject.setName(request.getParameter("ssub"));

try {

student.setStdId(sId); student.setName(sName) ; student.setsClass(sClass);
student.setUname(sUsername); student.setPassword(sPassword);
student.getSubject().setName(subject.getName());

subject.getStudent().add(student);

/* rs = -1 --> invalid subject.


                * rs =   0 --> Already registered.
                * rs =   1 --> Successfully registered.
                */
               int rs = addStudent.registerStudent(student);
               if( rs == -1)
               {
                    mv.addObject("msg", "Registration Fail :
Invalid Subject Name");
                    mv.setViewName("studentRegistrationResponse");
               }else if(rs == 0)
               {
                    mv.addObject("msg", "Registration Fail : Already
registered for the Test <a href = 'studentLogin.jsp'>go to login
page</a>");

                    mv.setViewName("studentRegistrationResponse");
               }else
               {
                    mv.addObject("msg", "Registration Success : <a
href = 'studentLogin.jsp'>go to login page</a>");
                    mv.setViewName("studentRegistrationResponse");
               }

          } catch (Exception e) {
               System.out.println("addStudent : "+e);
               mv.addObject("msg", "Unknown Error occured. Contact
Admin");
```

```
                setViewName("studentRegistrationResponse");
        mv.
            }
            return mv;
        }
}
```

MyWebInitializer.java

```
package com.sameer.HibernateSpringMVC;

/*
 * This class is used to map the servlet. Works as web.xml file
(Servlet mapping, configuring, etc.)
 */

import
org.springframework.web.servlet.support.AbstractAnnotationConfigDisp
atcherServletInitializer;


public class MyWebInitializer extends
AbstractAnnotationConfigDispatcherServletInitializer {


    @Override
    protected Class<?>[] getRootConfigClasses() {
        // TODO Auto-generated method stub return
        new Class[] {ProjectConfig.class};
    }
    @Override
    protected Class<?>[] getServletConfigClasses() {

        return new Class[] {ProjectConfig.class};
    }
    @Override
    protected String[] getServletMappings() {
        // TODO Auto-generated method stub
```

```
        return new String[] { "/" };
    }
}
```

ProjectConfig.java

```
package com.sameer.HibernateSpringMVC;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.view.InternalResourceViewResolver;

/* Request will come from MyWebInitializer
 * This class will forward request to servlet in specified package.
 * This file is replacement of sameer-servlet.java file of previous
project
 */
@Configuration
@ComponentScan({"com.sameer.HibernateSpringMVC"}) public
class ProjectConfig
{
    @Bean
    public InternalResourceViewResolver viewResolver()
    {
        InternalResourceViewResolver vr = new
InternalResourceViewResolver();

        vr.setSuffix(".jsp");

        return vr;
    }
```

}

Student.java

```
package com.sameer.HibernateSpringMVC.student.entities;

import java.util.ArrayList;
import java.util.List;

import     javax.persistence.Column;
import      javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.ManyToOne;

import com.sameer.HibernateSpringMVC.teacher.entities.Subject;

@Entity
public class Student {
        @Id
        private String stdId; private
        String name;
        @Column(name ="class")
        private String sClass; private
        double marks; private
        String uname;  private String
        password;

        @ManyToOne(targetEntity = Subject.class)
        private Subject subject = new Subject();

        public String getStdId() {
                return stdId;
        }

        public void setStdId(String stdId) {
                this.stdId = stdId;
        }

        public String getName() {
                return name;
        }

        public void setName(String name) {
                this. name = name;
        }

        public String getsClass() {
                return sClass;
        }
```

```
    public void setsClass(String sClass) {
        this.sClass = sClass;
    }

    public double getMarks() {
        return marks;
    }

    public void setMarks(double marks) {
        this.marks = marks;
    }

    public String getUname() {
        return uname;
    }

    public void setUname(String uname) {
        this.uname = uname;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public Subject getSubject() {
        return subject;
    }

    public void setSubject(Subject subject) {
        this.subject = subject;
    }
}
```

AddStudent.java

```
package com.sameer.HibernateSpringMVC.student; import

java.sql.SQLException;

import org.hibernate.Query; import
org.hibernate.Session;
import org.hibernate.SessionFactory; import
org.hibernate.Transaction; import
org.hibernate.cfg.Configuration;
import    org.hibernate.service.ServiceRegistry; import
org.hibernate.service.ServiceRegistryBuilder;

import com.sameer.HibernateSpringMVC.student.entities.Student; import
com.sameer.HibernateSpringMVC.teacher.entities.Subject;

public class AddStudent {
      public int registerStudent(Student student) throws
SQLException
      {
            Subject subject;

            /* rs = -1 --> invalid subject.
             * rs =   0 --> Already registered.
             * rs =   1 --> Successfully registered.
             */
            int rs = 1;

            Configuration conf = new
Configuration().configure("hibernate.cfg.xml").addAnnotatedClass(Stu
dent.class).addAnnotatedClass(Subject.class);
            ServiceRegistry reg = new
ServiceRegistryBuilder().applySettings(conf.getProperties()).buildSe
rviceRegistry();
            SessionFactory sf = conf.buildSessionFactory(reg); Session
            session = sf.openSession(); System.out.println("In Add
            Student");
            try {
                  Transaction tx = session.beginTransaction();
                  Query q = session.createQuery("from Subject where
name = :n");
                  q.setParameter("n", student.getSubject().getName());
                  subject = (Subject) q.uniqueResult();

                  if(subject.getsId().equals(null))
                  {
                        rs = -1;
                  }else
```

```
                {
                        student.getSubject().setsId(subject.getsId());
                        session.save(student);
                        tx.commit();

                }

        } catch (org.hibernate.exception.ConstraintViolationException
e) {
                rs = 0;
                System.out.println("From AddStudent : "+e);
        }catch(java.lang.NullPointerException e) { rs
                = -1;
                System.out.println("From AddStudent : "+e);
        }

        return rs;
    }
}
```

Subject.java

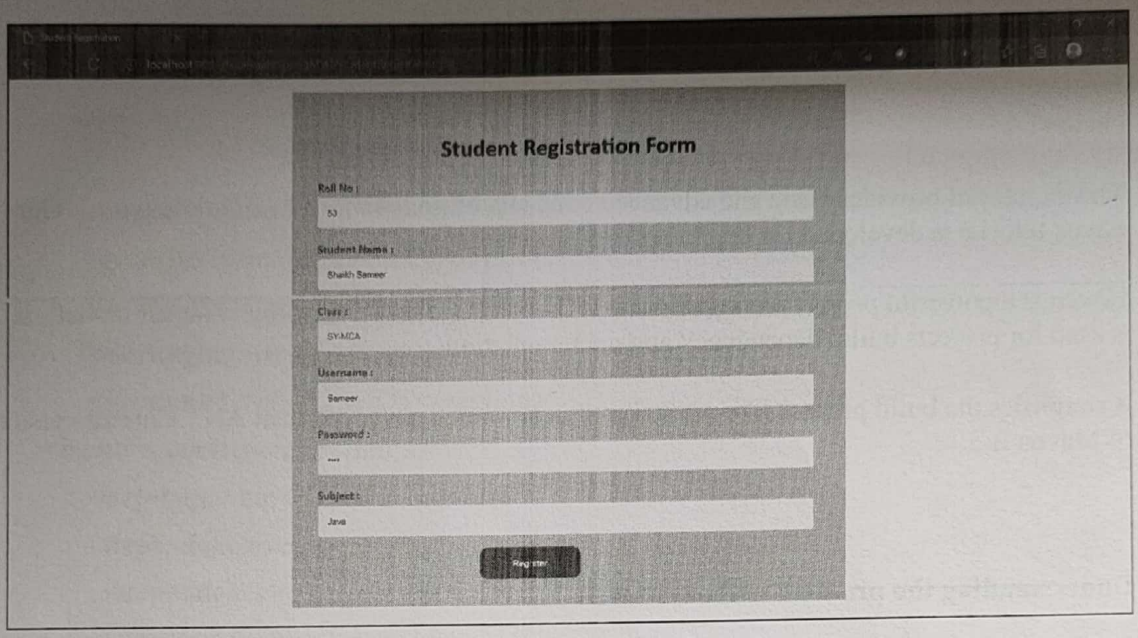
```
package com.sameer.HibernateSpringMVC.teacher.entities;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;

import com.sameer.HibernateSpringMVC.student.entities.Student;


@Entity
public class Subject {
    @Id
    private String sId;
    private String name;
    @OneToMany(mappedBy="subject")
    private List<Student> student = new ArrayList<Student>();


    public String getsId() {
        return sId;
    }
    public void setsId(String sId) {
        this.sId = sId;
```

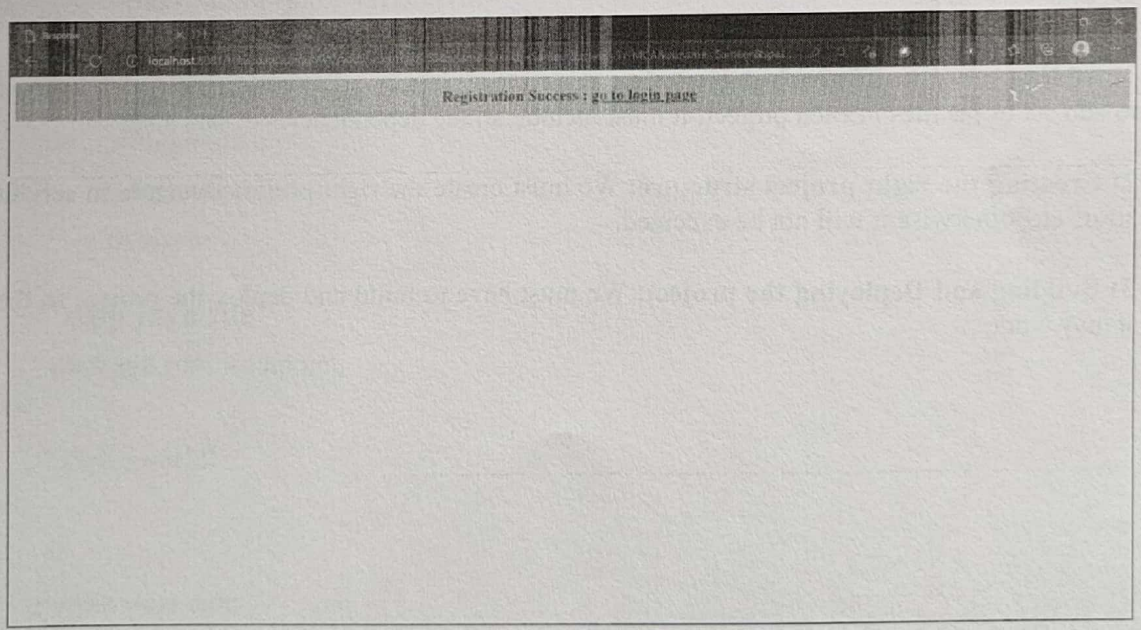
```
        }
        public String getName() {
                return name;
        }
        public void setName(String name) {
                this.name = name;
        }
        public List<Student> getStudent() {
                return student;
        }
        public void setStudent(List<Student> student) {
                this.student = student;
        }
}
```

Hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
                "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
                "http://www.hibernate.org/dtd/hibernate-configuration-
                3.0.dtd">
        <hibernate-
    configuration>
   <session-factory>
          <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
          <property name="hibernate.connection.password">root</property>
          <property
name="hibernate.connection.url">jdbc:mysql:///localhost:3306/college_test
</property
>
          <property name="hibernate.connection.username">sameer</property>
          <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

          <!-- We are telling Hibernate to create table
                for me create -> Create new table.
               update -> Update existing table. If not exists create table. -
               ->
          <property name="hbm2ddl.auto">update</property>
          <!-- We are trying to print the sql queries performed by Hibernate
          -->
          <property name="show_sql">true</property>

   </session-factory>
      </hibernate-
     configuration>
```

-: Output :-





Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 10

Aim: Write a program for implementing concept of Maven Project.

Theory:

Maven tutorial provides basic and advanced concepts of **apache maven** technology. Our maven tutorial is developed for beginners and professionals.

Maven is a powerful *project management tool* that is based on POM (project object model). It is used for projects build, dependency and documentation.

It simplifies the build process like ANT. But it is too much advanced than ANT.Current version of Maven is 3.

Understanding the problem without Maven

There are many problems that we face during the project development. They are discussed below:

1) Adding set of Jars in each project: In case of struts, spring, hibernate frameworks, we need to add set of jar files in each project. It must include all the dependencies of jars also.

2) Creating the right project structure: We must create the right project structure in servlet, struts etc, otherwise it will not be executed.

3) Building and Deploying the project: We must have to build and deploy the project so that it may work.

Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 10

Aim: Write a program for implementing concept of Maven Project.

Theory:

Maven tutorial provides basic and advanced concepts of **apache maven** technology. Our maven tutorial is developed for beginners and professionals.

Maven is a powerful *project management tool* that is based on POM (project object model). It is used for projects build, dependency and documentation.

It simplifies the build process like ANT. But it is too much advanced than ANT.Current version of Maven is 3.

Understanding the problem without Maven

There are many problems that we face during the project development. They are discussed below:

1) Adding set of Jars in each project: In case of struts, spring, hibernate frameworks, we need to add set of jar files in each project. It must include all the dependencies of jars also.

2) Creating the right project structure: We must create the right project structure in servlet, struts etc, otherwise it will not be executed.

3) Building and Deploying the project: We must have to build and deploy the project so that it may work.

Program :

pom.xml file

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
 http://maven.apache.org/maven-v4_0_0.xsd">

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.javatpoint</groupId>
  <artifactId>CubeGenerator</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
<name>CubeGenerator</name>
  <url>http://maven.apache.org</url>
  <dependencies>
   <dependency>
    <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>3.8.1</version>
     <scope>test</scope>
    </dependency>
   </dependencies>
  </project>
```

App.java file

```
 package com.javatpoint;
/**
 * Hello world!
*
 */
 public class App
 {
public static void main( String[] args )
{
 System.out.println( "Hello World!" );    }
 }
```

AppTest.java

```
package com.javatpoint;

import junit.framework.Test;
import junit.framework.TestCase;
import junit.framework.TestSuite;
/**
 * Unit test for simple App.
 */
public class AppTest
    extends TestCase
{
    /**
     * Create the test case
     *
     * @param testName name of the test case
     */
    public AppTest( String testName )
    {
        super( testName );
    }
    /**
     * @return the suite of tests being tested
     */
    public static Test suite()
    {
        return new TestSuite( AppTest.class );
    }
    /**
     * Rigourous Test :-)
     */
    public void testApp()
    {
        assertTrue( true );
    }
}
```

Run the Maven Java Project

java com.javatpoint.App

Output of the maven example

Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 11

Aim: Write a program for implementing concept of Web Service.

Theory:

Web Services Tutorial

Web Services tutorial is designed for beginners and professionals providing basic and advanced concepts of web services such as protocols, SOAP, RESTful, java web service implementation, JAX-WS and JAX-RS tutorials and examples.

Web service is a technology to communicate one programming language with another. For example, java programming language can interact with PHP and .Net by using web services. In other words, web service provides a way to achieve interoperability.

Java Web Services Tutorial

In this tutorial, you will be able to learn java web services and its specifications such as JAX-WS and JAX-RS.

There are two ways to write the code for JAX-WS by RPC style and Document style. Like JAX-WS, JAX-RS can be written by Jersey and RESTeasy. We will learn all these technologies later.

What are the Different Types of Web Services?

There are a few central types of web services: XML-RPC, UDDI, SOAP, and REST:

XML-RPC (Remote Procedure Call) is the most basic XML protocol to exchange data between a wide variety of devices on a network. It uses HTTP to quickly and easily transfer data and communication other information from client to server.

UDDI (Universal Description, Discovery, and Integration) is an XML-based standard for detailing, publishing, and discovering web services. It's basically an internet registry for businesses around the world. The goal is to streamline digital transactions and e-commerce among company systems.

SOAP, which will be described in detail later in the blog, is an XML-based Web service protocol to exchange data and documents over HTTP or SMTP (Simple Mail Transfer Protocol). It allows independent processes operating on disparate systems to communicate using XML.

REST, which will also be described in great detail later in the blog, provides communication and connectivity between devices and the internet for API-based tasks. Most RESTful services use HTTP as the supporting protocol.

Here are some well-known web services that use markup languages:

- Web template
- JSON-RPC
- JSON-WSP
- Web Services Description Language (WSDL)
- Web Services Conversation Language (WSCL)
- Web Services Flow Language (WSFL)
- Web Services Metadata Exchange (WS-MetadataExchange)
- XML Interface for Network Services (XINS)

Program :

Web.xml

```
<?xml version="1.0"?>
<web-app
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
version="2.4">

<description>Employee lookup</description>
<display-name>Employee-lookup</display-name>

<servlet>
  <servlet-name>XQueryWebService</servlet-name>
  <servlet-class>com.ddtek.xquery.webservice.XQServlet
  </servlet-class>
  <init-param>
    <param-name>JNDIDataSourceName</param-name>
    <param-value>jdbc/employee-lookup</param-value>
  </init-param>
  <init-param>
    <param-name>QueriesFolder</param-name>
    <param-value>c:\MyQueryDir</param-value>
  </init-param>
</servlet>
<resource-ref>
  <res-ref-name>jdbc/employee-lookup</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
<servlet-mapping>
  <servlet-name>XQueryWebService</servlet-name>
  <url-pattern>/*</url-pattern>
</servlet-mapping>
</web-app>
```

CREATING A CONNECTION POOL

Here's how to create a connection pool in Apache Tomcat:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Context path="/employee-lookup" docBase="employee-lookup"
crossContext="false" reloadable="true" debug="0">
<Resource name="jdbc/employee-lookup"
   auth="Container"
   type="javax.sql.DataSource"
   username="root"
   password="sa"
   driverClassName="com.ddtek.xquery3.jdbc.XQueryDriver"
url="jdbc:datadirect:xquery3://JdbcUrl=
{jdbc:mysql://localhost:3306/pubs_dbo?}"
   initialSize="1"
   accessToUnderlyingConnectionAllowed="true"
   validationQuery="SELECT * FROM users"/>
</Context>
```

Using SOAP

Using SOAP, on the other hand, requires submitting the following SOAP request (XML):

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <dd:emp xmlns:dd="http://www.datadirect.com">
         <dd:id>A-C71970F</dd:id>
      </dd:emp>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

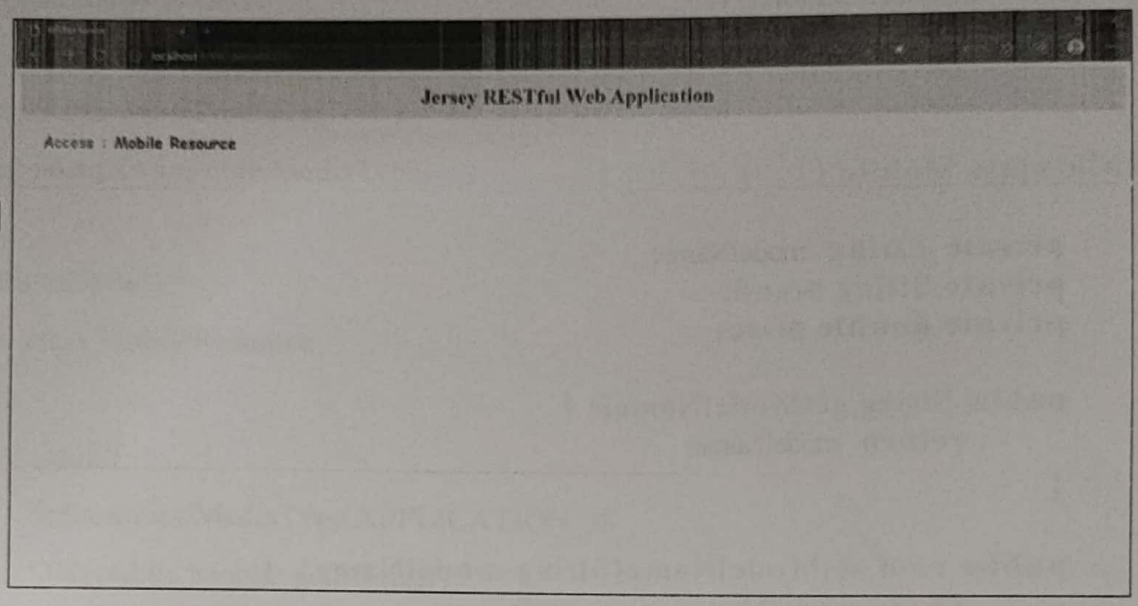
MobileResource.java

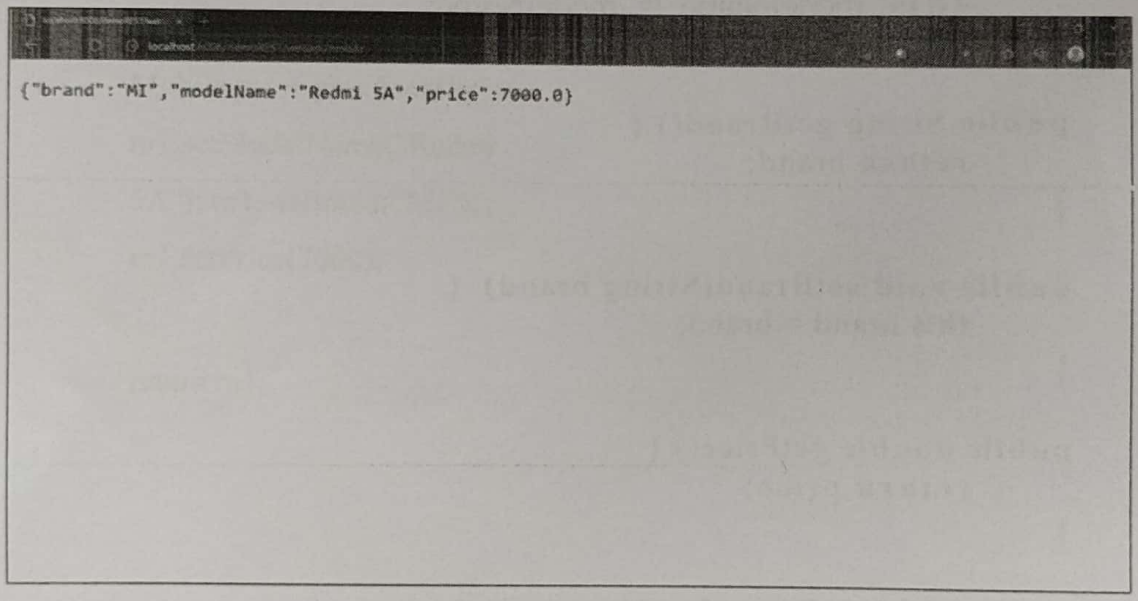
```
import jakarta.ws.rs.GET;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.Produces;
import jakarta.ws.rs.core.MediaType;

@Path("mobile")
public class MobileResource
{
        @GET
        @Produces(MediaType.APPLICATION_JS
        ON) public Mobile getMobile()
        {
                Mobile m1 = new Mobile();
                m1.setModelName("Redmi
                5A"); m1.setBrand("MI");
                m1.setPrice(7000);

                return m1;
        };

}
```

Mobile.java

```
package com.sameer.demoREST;
import jakarta.json.bind.annotation.JsonbAnnotation;

public class Mobile {

	private String modelName;
	private String brand;
	private double price;

	public String getModelName() {
		return modelName;
	}

	public void setModelName(String modelName) {
		this.modelName = modelName;
	}

	public String getBrand() {
		return brand;
	}

	public void setBrand(String brand) {
		this.brand = brand;
	}

	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

}
```

-: Output :-





Class: SY(MCA) **Subject: Lab I- Advance Java**

Experiment No. 12

Aim: Write a program for implementing concept of Junit Service.

Theory:

JUnit Tutorial | Testing Framework for Java

JUnit tutorial provides basic and advanced concepts of **unit testing in java** with examples. Our junit tutorial is designed for beginners and professionals.

It is an *open-source testing framework* for java programmers. The java programmer can create test cases and test his/her own code.

It is one of the unit testing framework. Current version is junit 4.

To perform unit testing, we need to create test cases. The **unit test case** is a code which ensures that the program logic works as expected.

Types of unit testing

There are two ways to perform unit testing: 1) manual testing 2) automated testing.

1) Manual Testing

If you execute the test cases manually without any tool support, it is known as manual testing. It is time consuming and less reliable.

2) Automated Testing

If you execute the test cases by tool support, it is known as automated testing. It is fast and more reliable.

Program :

```
package com.javatpoint.logic;
public class Calculation {
//method that returns maximum number
public static int findMax(int arr[]){
    int max=0;
    for(int i=1;i<arr.length;i++){
        if(max<arr[i])
            max=arr[i];
    }
    return max;
}
//method that returns cube of the given number
public static int cube(int n){
    return n*n*n;
}
//method that returns reverse words
public static String reverseWord(String str){

    StringBuilder result=new StringBuilder();
    StringTokenizer tokenizer=new StringTokenizer(str," ");

    while(tokenizer.hasMoreTokens()){
    StringBuilder sb=new StringBuilder();
    sb.append(tokenizer.nextToken());
    sb.reverse();

    result.append(sb);
    result.append(" ");
    }
    return result.toString();
  }
}
```

Write the test case

```
package com.javatpoint.testcase;

import static org.junit.Assert.assertEquals;
import org.junit.After;
import org.junit.AfterClass;
import org.junit.Before;
import org.junit.BeforeClass;
import org.junit.Test;
import com.javatpoint.logic.Calculation;

public class TestCase2 {

  @BeforeClass
  public static void setUpBeforeClass() throws Exception {
    System.out.println("before class");
  }
  @Before
  public void setUp() throws Exception {
    System.out.println("before");
  }

  @Test
  public void testFindMax(){
    System.out.println("test case find max");
    assertEquals(4,Calculation.findMax(new int[]{1,3,4,2}));
    assertEquals(-2,Calculation.findMax(new int[]{-12,-3,-4,-2}));
  }
  @Test
  public void testCube(){
    System.out.println("test case cube");
    assertEquals(27,Calculation.cube(3));
  }
  @Test
  public void testReverseWord(){
    System.out.println("test case reverse word");
    assertEquals("ym eman si nahk",Calculation.reverseWord("my name is khan");
```

```
    }
    @After
    public void tearDown() throws Exception {
        System.out.println("after");
    }

    @AfterClass
    public static void tearDownAfterClass() throws Exception {
        System.out.println("after class");
    }

}
```

OUTPUT: