**RAJASTHAN INSTITUTE OF ENGINEERING AND TECHNOLOGY, JAIPUR**

**Bachelor of Technology**

**(in Department of Computer Science &Engineering)**



**Rajasthan Institute of Engineering & Technology, Jaipur**

**Rajasthan Technical University, Kota**

**Session - 2018-2022**

**ADVANCE JAVA LAB (5CS4-22)**

**Submitted to:- Submitted by:**

**Ms. Shalini Sharma Ankit Malpani**

**(Assistant professor) Roll No.: 18ERECS009**

**(CSE DEPARTMENT) Batch – A1**

**CSE (3rd year)**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Experiment Name** | **Page No.** |
| **1.** | Write a program for helloworld using Swing and JFrame. | 3-4 |
| **2.** | Write a program for Creating a Main Frame. | 5-6 |
| **3.** | Write a program for Creating a Panel. | 7-8 |
| **4.** | Write a program for Creating a Panel class. | 9-10 |
| **5.** | Write a program for Check Box. | 11 |
| **6.** | Write a program for Password Field. | 12 |
| **7.** | Write a program for Combo Box. | 13 |
| **8.** | Write a program for Creating Table. | 14 |
| **9.** | Write a program for Creating Radio Button. | 15 |
| **10.** | Write a program for Database Connectivity. | 16 |
| **11.** | Write a program for Protocol Handing. | 17-18 |
| **12.** | Write a program for Socket Programming. | 19-20 |
| **13.** | Write a program for Content Handing. | 21-23 |
| **14.** | Write a program for Creating a JSP(Java Sever Page) file. | 24-26 |

# Experiment No. 1

**Aim**:Write a program for helloworld using Swing and JFrame.

**Resources:**JDK 11, eclipse.

**Description:**

* 1. **Swing:** Swing in java is part of Java foundation class which is lightweight and platform independent. It is used for creating window based applications. It includes components like button, scroll bar, text field etc. Putting together all these components makes a graphical user interface. In this article, we will go through the concepts involved in the process of building applications using swing in[Java.](https://www.edureka.co/java-j2ee-training-course)
  2. **JFrame:** The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works like the main window where components like labels, buttons, textfields are added to create a GUI.

**Program:**

import javax.swing.JFrame;

import javax.swing.SwingUtilities;

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

SwingUtilities.invokeLater(new Runnable() { public void run() {

JFrame frame = new JFrame("Hello world swing");

frame.setVisible(true); frame.setSize(500, 500);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

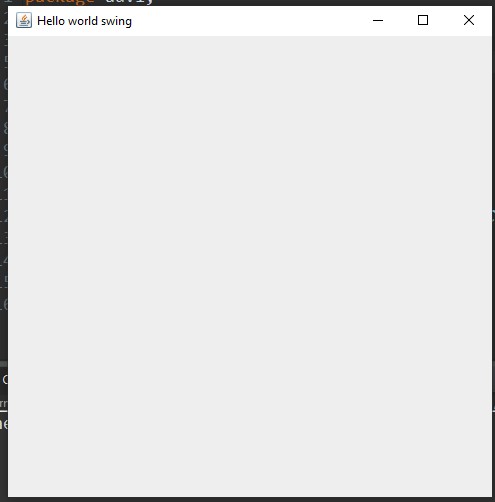
}

});

}

}

**Output:**



# Experiment No. 2

**Aim** : Write a program for Creating a Main Frame.

**Resources:**JDK 11, eclipse.

**Description:**

**a.MainFrame:**MainFrame class which holds the code for designing our JFrame. Inside the MainFrame class there is a constructor which creates our JFrame object when it is instantiated, so it is only natural that the code for designing the JFrame goes inside the constructor and gets created when instantiated.

**Program:**

import java.awt.BorderLayout; import java.awt.Container; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JTextArea; public class MainFrame extends JFrame { public MainFrame(String title) { super(title);

// Set layout Manager

setLayout(new BorderLayout());

//Create Swing component

JTextAreatextArea = new JTextArea();

JButton button = new JButton("Click me");

//Add Swing components to content pane

Container c = getContentPane();

c.add(textArea, BorderLayout.CENTER);

c.add(button, BorderLayout.SOUTH); //Add behavior

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

textArea.append("Hello");

}

});

}

}

**Output:**



# Experiment No. 3

**Aim:** Write a program for Creating a Panel.

**Resources:**JDK 11, eclipse.

**Description:**

**a.Panel:**The Panel is a simplest container class. It provides space in which an application can attach any other component. It inherits the Container class.

**Program:**

import java.awt.BorderLayout; import java.awt.Container; import java.awt.event.ActionEvent; import java.awt.event.ActionListener; import javax.swing.JButton; import javax.swing.JFrame; import javax.swing.JTextArea; public class MainFrame extends JFrame { private DetailsPaneldetailsPanel; public MainFrame(String title) { super(title);

// Set layout Manager setLayout(new BorderLayout());

//Create Swing component

JTextAreatextArea = new JTextArea(); JButton button = new JButton("Click me"); detailsPanel = new DetailsPanel();

//Add Swing components to content pane Container c = getContentPane();

c.add(textArea, BorderLayout.CENTER);

c.add(button, BorderLayout.SOUTH);

c.add(detailsPanel, BorderLayout.WEST);

//Add behavior

button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

textArea.append("Hello");

}

});

}

}

**Output:**



# Experiment No. 4

**Aim** :Write a program for Creating a Panel class.

**Resources:**JDK 11, eclipse.

**Description:**

**a.panel:**The Panel is a simplest container class. It provides space in which an application can attach any other component. It inherits the Container class.

**Program:**

import java.awt.Dimension;

import java.awt.GridBagConstraints; import java.awt.GridBagLayout;

import java.awt.Panel;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JTextField;

public class DetailsPanel extends Panel { public DetailsPanel() {

Dimension size = getPreferredSize(); size.width = 200; setPreferredSize(size);

JLabelnameLabel = new JLabel("Name:");

JLabeloccupationalLabel = new JLabel("Occupation:");

JTextFieldnameField = new JTextField(10);

JTextFieldoccupationalField = new JTextField(10);

JButtonaddBtn = new JButton("Add"); setLayout(new GridBagLayout());

GridBagConstraintsgc = new GridBagConstraints();

}

}

# GridBag Constraints

// First column//

gc.anchor = GridBagConstraints.LAST\_LINE\_END; gc.weightx = 0.5; gc.weighty = 0.5; gc.gridx = 0; gc.gridy = 0;

add(nameLabel, gc);

gc.gridx = 0; gc.gridy = 1; add(occupationalLabel, gc);

// Second Column

gc.anchor = GridBagConstraints.LAST\_LINE\_START; gc.gridx = 1; gc.gridy = 0; add(nameField, gc);

gc.gridx = 1; gc.gridy = 1; add(occupationalField, gc);

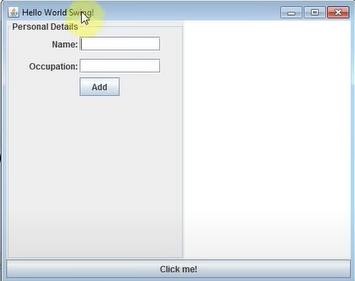
// Final row

gc.weightx = 10;

gc.anchor = GridBagConstraints.FIRST\_LINE\_START; gc.gridx = 1; gc.gridy = 2;

add(addBtn, gc);

**Output:**



# Experiment No. 5

**Aim**: Write a program for Check Box.

**Resources:**JDK 11, eclipse.

**Description:**

**a.Check Box:** The Checkbox class is used to create a checkbox. It is used to turn an option on (true) or off (false). Clicking on a Checkbox changes its state from "on" to "off" or from "off" to "on".

**Program:**

import javax.swing.\*;

public class jpac5

{

jpac5(){

JFrame f= new JFrame("CheckBox Example");

JCheckBox checkBox1 = new JCheckBox("C++");

checkBox1.setBounds(100,100, 50,50);

JCheckBox checkBox2 = new JCheckBox("Java", true);

checkBox2.setBounds(100,150, 50,50);

f.add(checkBox1);

f.add(checkBox2);

f.setSize(400,400);

f.setLayout(null);

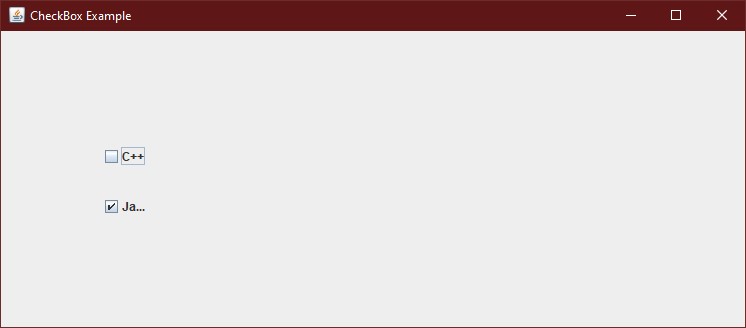
f.setVisible(true);

}

public static void main(String args[]){

new jpac5();}}

**Output:**



**Experiment No. 6**

**Aim:**Write a program for Password Field.

**Resources:**JDK 11, eclipse.

**Description:**

**a.Password Field:** The object of a JPasswordField class is a text component specialized for password entry. It allows the editing of a single line of text. It inherits JTextField class.

**Program:**

import javax.swing.\*;

public class jpac6 {

public static void main(String[] args) {

JFrame f=new JFrame("Password Field Example");

JPasswordField value = new JPasswordField();

JLabel l1=new JLabel("Password:");

l1.setBounds(20,100, 80,30);

value.setBounds(100,100,100,30);

f.add(value); f.add(l1);

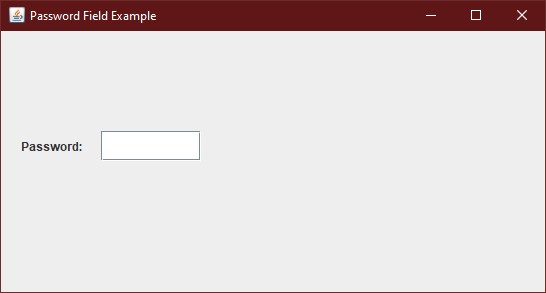
f.setSize(300,300);

f.setLayout(null);

f.setVisible(true);

}}

**Output:}**



# Experiment No. 7

**Aim**: Write a program for Combo Box.

**Resources:**JDK 11, eclipse.

**Description:**

**a.Combo Box:** The object of Choice class is used to show popup menu of choices. Choice selected by user is shown on the top of a[menu.](https://www.javatpoint.com/java-jmenuitem-and-jmenu) It inherits[JComponent](https://www.javatpoint.com/java-jcomponent)class.

**Program:**

import javax.swing.\*;

public class jpac7{

JFrame f;

jpac7(){

f=new JFrame("ComboBox Example");

String country[]={"India","Aus","U.S.A","England","Newzealand"};

JComboBox cb=new JComboBox(country);

cb.setBounds(50, 50,90,20);

f.add(cb);

f.setLayout(null);

f.setSize(400,500);

f.setVisible(true);

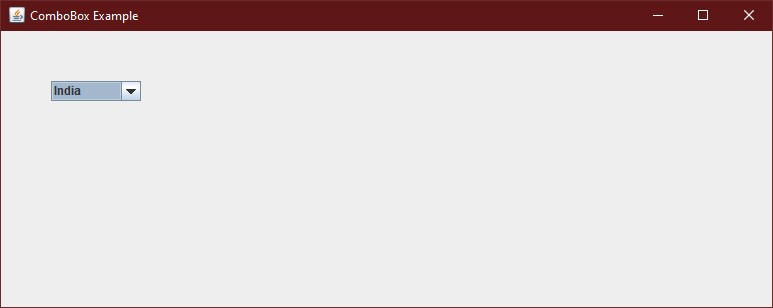
}

public static void main(String[] args) {

new jpac7();

}}

**Output:**



# Experiment No. 8

**Aim:**Write a program for Creating Table.

**Resources:**JDK 11, eclipse.

**Description:**

**a.Creating Table:** The JTable class is used to display data in tabular form. It is composed of rows and columns.

**Program:**

import javax.swing.\*;

public class jpac8 {

JFrame f;

jpac8(){

f=new JFrame();

String data[][]={ {"1","Ankit","235346"}, {"2","vicy","780000"},

{"3","sfd","700000"}};

String column[]={"ID","NAME","SALARY"};

JTable jt=new JTable(data,column);

jt.setBounds(30,40,200,300);

JScrollPane sp=new JScrollPane(jt);

f.add(sp);

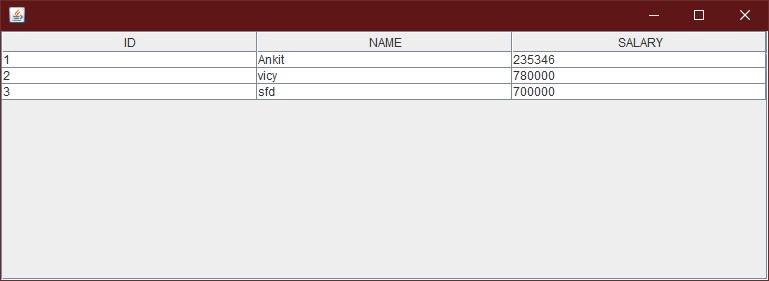
f.setSize(300,400);

f.setVisible(true);}

public static void main(String[] args) {

new jpac8();}}

**Output:**



**Experiment No. 9**

**Aim** : Write a program for Creating Radio Button.

**Resources:**JDK 11, eclipse.

**Description:**

**a.Radio Button:** The JRadioButton class is used to create a radio button. It is used to choose one option from multiple options. It is widely used in exam systems or quiz.

**Program:**

import javax.swing.\*;

public class jpac9 {

JFrame f;

jpac9(){f=new JFrame();

JRadioButton r1=new JRadioButton("A) No-1");

JRadioButton r2=new JRadioButton("B) No-2");

JRadioButton r3=new JRadioButton("C) No-3");

r1.setBounds(75,50,100,30); r2.setBounds(75,100,100,30);r3.setBounds(75,150,100,30);

ButtonGroup bg=new ButtonGroup();

bg.add(r1);bg.add(r2);bg.add(r3);

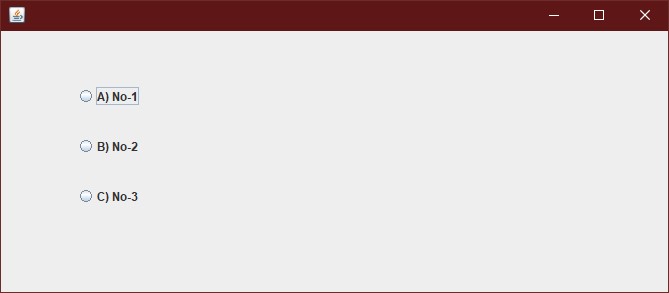
f.add(r1);f.add(r2);f.add(r3);

f.setSize(300,300);

f.setLayout(null);

f.setVisible(true);}

public static void main(String[] args) {new jpac9();}}

**Output**

**Experiment No. 10**

**Aim :** Write a program for Database Connectivity.

**Resources :** JDK 11, eclipse.

**Description:**

**a.JDBC:** Java Database Connectivity is an application programming interface for the programming language Java, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. It is part of the Java Standard Edition platform, from Oracle Corporation.

**Program:**

import java.sql.\*; class MysqlCon{

public static void main(String args[]){ try{

Class.*forName*("com.mysql.jdbc.Driver");

Connection conn =

DriverManager.*getConnection*("jdbc:mysql://localhost:3306/test",

"root", "12345678");

Statement stmt=con.createStatement();

ResultSetrs=stmt.executeQuery("select \* from user"); while(rs.next())

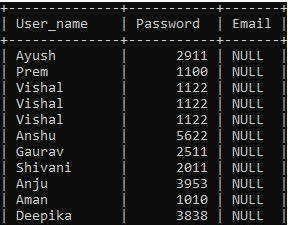
System.out.println(rs.getInt(1)+""+rs.getString(2)+""+rs.getString(3)

);

con.close(); } catch(Exception e){

System.out.println(e); }}}

**Output:**



# Experiment No. 11

**Aim :** Write a program for Protocol Handing.

**Resources:**JDK 11, eclipse.

**Description:**

**Protocol Handing:** Java divides the task of handlingprotocols into a number of pieces. As a result, there is no single class called ProtocolHandler . Instead, pieces of the protocolhandler mechanism are implemented by four different class es in the java.net package: URL , URLStreamHandler , URLConnection , and URLStreamHandlerFactory .

**Program:**

import java.net.\*; import java.io.\*; public class Pro2{

public static void main(String [] args) {

try {

URL url = new

URL("https://www.amrood.com/index.htm?language=en#j2se");

System.out.println("URL is " + url.toString());

System.out.println("protocol is " + url.getProtocol());

System.out.println("authority is " + url.getAuthority());

System.out.println("file name is " + url.getFile());

System.out.println("host is " + url.getHost());

System.out.println("path is " + url.getPath());

System.out.println("port is " + url.getPort());

System.out.println("default port is " + url.getDefaultPort());

System.out.println("query is " + url.getQuery());

System.out.println("ref is " + url.getRef());

}

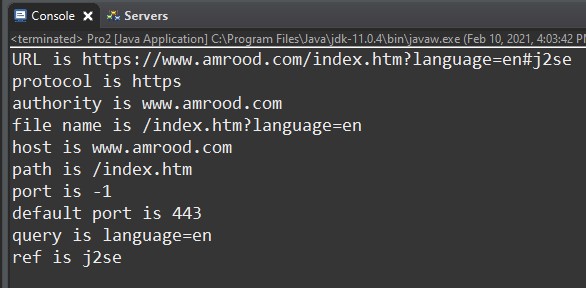
catch (IOException e) {

e.printStackTrace();

}

}}

**Output:**



# Experiment No. 12

**Aim :** Write a program for Socket Programming.

**Resources:**JDK 11, eclipse.

**Description:**

* 1. **Socket Programming:** This article describes a very basic one-way Client and Server setup where a Client connects, sends messages to server and the server shows them using socket connection. There’s a lot of low-level stuff that needs to happen for these things to work but the Java API networking package (java.net) takes care of all of that, making network programming very easy for programmers.
  2. **Client SideProgramming:**It is the program that runs on the client machine (browser) and deals with the user interface/display and any other processing that can happen on client machine like reading/writing cookies.
  3. **Server SideProgramming:**It is the general name for the kind of program that runs directly on the server.Or we can say that server-side programming must deal with dynamic content. It runs on the server. Most web pages are not static since they deal with searching databases.

**Program:**

* 1. **Client Side Program:**

import java.io.\*; import java.net.\*; public class Cli {

public static void main(String[] args){ try{

ServerSocket ss = new ServerSocket(6666);

Socket s=ss.accept();

DataInputStream dis = new DataInputStream(s.getInputStream());

String str = (String)dis.readUTF(); System.out.println("message=" +str); ss.close();

} catch(Exception e){ System.out.println(e);

}}

**b.Server Side Program:**

import java.io.\*;

import java.io.DataOutputStream;

import java.net.\*; public class Ser {

public static void main(String[] args) { try{

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

DataOutputStream(s.getOutputStream()); dout.writeUTF("HelloServer"); dout.flush(); dout.close(); s.close();

}

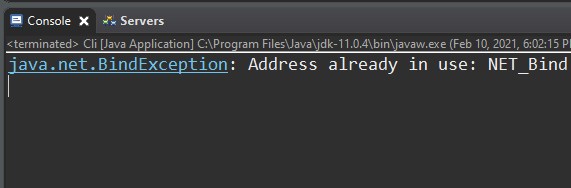
catch(Exception e){

System.*out*.println(e);

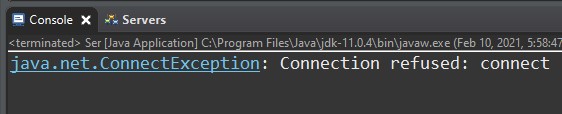
}}}

**Output:**

1. **Output for Client Side:**



1. **Output for Client Side:**



# Experiment No. 13

**Aim:** Write a program for Content Handing.

**Resources:**JDK 11, eclipse.

**Description:**

* 1. **Creating a File:** To create a file in Java, you can use the createNewFile() method. This method returns a boolean value: true if the file was successfully created, and false if the file already exists. Note that the method is enclosed in a try...catch block. This is necessary because it throws an IOException if an error occurs (if the file cannot be created for some reason).
  2. **File Methods:**The File class is an abstract representation of file and directory pathname. A pathname can be either absolute or relative.The File class have several methods for working with directories and files such as creating new directories or files, deleting and renaming directories or files, listing the contents of a directory etc.
  3. **Deleting a File:** Java provides methods to delete files using java programs. On the contrary to normal delete operations in any operating system, files being deleted using java program is deleted permanently without being moved to trash/recycle bin.

**Program:**

* 1. **Program for Creating a File:**

import java.io.File; public class def {

public static void main(String[] args) { File f = null; String[] strs = {

"java1.txt",

"java2.txt",

"java3.txt",

"DataFlair.txt"

}; try {

for (String s: strs) { f = new File(s); boolean bool = f.canExecute();

String a = f.getAbsolutePath();

System.*out*.println(a + " executable? " + bool);

}

}

catch(Exception e) {

e.printStackTrace();

}

}

**b.Program for File Methods:**

import java.io.File; public class def {

public static void main(String[] args) {

File f = null;

String[] strs = {

"java1.txt",

"java2.txt",

"java3.txt",

"DataFlair.txt"

}; try {

for (String s: strs) { f = new File(s); boolean bool = f.canExecute();

String a = f.getAbsolutePath();

System.*out*.println(a + " executable? " + bool);

}

}

catch(Exception e)

{

e.printStackTrace();

}}}

**c.Program for Deleting a File:**

import java.io. \* ; public class del { public static void main(String[] args) {

File file = new File("C:/internshipDF/File Class/java1.txt"); boolean delete = file.delete();

if(delete) {

System.*out*.println("The file is no more");

}

else {

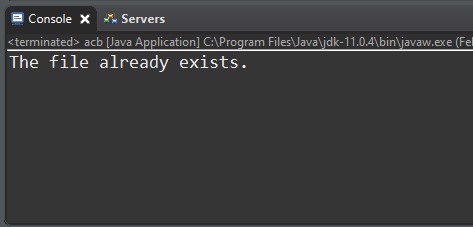
System.*out*.println("The file was not found!");

}

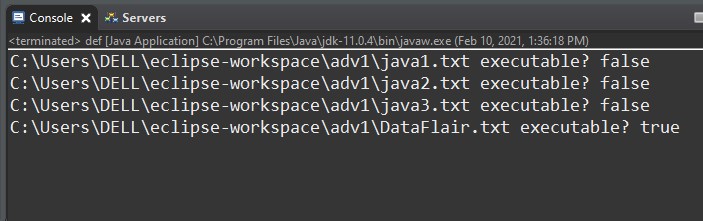
}

**Output:**

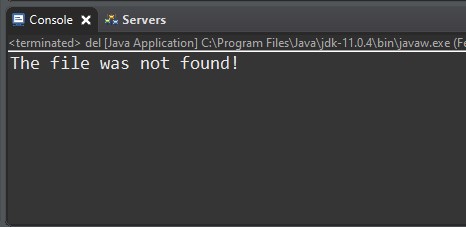
1. **Output for Creating a File:**



1. **Output for File Methods:**



**c .Output for Deleting a File:**



# Experiment No. 14

**Aim:** Write a program for Creating a JSP(Java Sever Page) file.

**Resources:** JDK 11, eclipse, Chrome Browser, html.

**Description:**

**a.JSP (Java Server Page):** JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

**Program:**

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>JSP Page</title>

</head>

<body>

<p> Hi, <h3><%="Hello!" %></h3></p>

<p>You successfully run a simple jsp program.</p>

<p>Learn More concept about jsp.</p>

</body>

</html>

<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Registration Form</title>

</head>

<body>

<h1>Register Form</h1>

<form action="guru\_register" method="post">

<table style="with: 50%">

<tr>

<td>First Name</td>

<td><input type="text" name="first\_name" /></td>

</tr>

<tr>

<td>Last Name</td>

<td><input type="text" name="last\_name" /></td>

</tr>

<tr>

<td>UserName</td>

<td><input type="text" name="username" /></td>

</tr>

<tr>

<td>Password</td>

<td><input type="password" name="password" />

</td>

</tr>

<tr>

<td>Address</td>

<td><input type="text" name="address" /></td>

</tr>

<tr>

<td>Contact No</td>

<td><input type="text" name="contact" /></td>

</tr>

</table>

<input type="submit" value="Submit" /></form>

</body>

</html>

**Output:-**

