Table of Contents

[Lab 01: A Java program to create a registration form using the concept of Swing, Event Handling and File Handling. 1](#_Toc132702354)

[LAB 02: A java program to create a Login Form using the concept of Swing, Event Handling and File Handling. 7](#_Toc132702355)

[LAB 03: A java program to add a content in a table using the concept of Swing and Event Handling. 12](#_Toc132702356)

[LAB 04: A program to create a simple calculator GUI using the concept of Swing, Event Handling in Java programming Language. 20](#_Toc132702357)

[LAB 05A: A program using TCP such that client sends number to server and displays its factorial. The server computes factorial of the number received from client. 26](#_Toc132702358)

[LAB 05B: A java program using UDP showing that the sending and receiving of message using Datagram Packet and Datagram Socket class. 30](#_Toc132702359)

[LAB 06: You are hired by a reputed software company which is going to design an application for "Movie Rental System". Your responsibility is to design a schema named MRS and create a table named Movie(id, Title, Genre, Language, Length). Write a program to design a GUI form to take input for this table and insert the data into table after clicking the OK button. 33](#_Toc132702360)

[LAB 07: A servlet application to print current date and time, communicate between html and servlet and to create an auto refreshed page. 38](#_Toc132702361)

[**7.1 To print current date and time:** 38](#_Toc132702362)

[**7.2 Communicate between Html and servlet.** 42](#_Toc132702363)

[**7.3 Create an auto refreshed page.** 45](#_Toc132702364)

[LAB 08: A JSP application program to login. 50](#_Toc132702365)

[LAB 09: RMI program to add two number. 54](#_Toc132702366)

# Lab 01: A Java program to create a registration form using the concept of Swing, Event Handling and File Handling.

**OBJECTIVE:** To create a registration form using the concept of Swing, Event Handling and File Handling in Java programming language.

Theory:

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. Swing was created to provide more powerful and flexible components than Java AWT (Abstract Window Toolkit).The Swing GUI components are defined in the javax.swing package.

Methods Used:

1. setVisible(boolean b):Shows or hides a component; it is false by default
2. add(Component c):Adds a component
3. setLayout(LayoutManager m):Sets a layout manager
4. setSize(int w, int h):Sizes the component
5. setBackground(Color c):sets the background color of the specified container
6. addActionListenerListener(ActionListener d): add actionListener for specified component

**Source Code:**

import java.awt.\*;

import java.awt.event.\*;

import java.io.FileOutputStream;

import javax.swing.\*;

public class Registration extends JFrame implements ActionListener{

private Container C;

private JLabel l1, l2, l3, l4, l5;

private JTextField t1, t2, t3;

private JRadioButton b1, b2, b3;

private JComboBox cb1, cb2, cb3;

private JCheckBox c1;

private JButton submit;

private String gender, selectedDay, selectedMth, selectedYear;

public Registration(){

setTitle("Registration");

setSize(500, 400);

setLocationRelativeTo(null);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

C=this.getContentPane();

C.setLayout(null);

l1=new JLabel("Name");

l1.setBounds(50, 30, 100, 30);

C.add(l1);

t1=new JTextField();

t1.setBounds(150, 30, 300, 30);

C.add(t1);

l2=new JLabel("Email Address");

l2.setBounds(50, 60, 100, 30);

C.add(l2);

t2=new JTextField();

t2.setBounds(150, 60, 300, 30);

C.add(t2);

l3=new JLabel("Phone Number");

l3.setBounds(50, 90, 100, 30);

C.add(l3);

t3=new JTextField();

t3.setBounds(150, 90, 300, 30);

C.add(t3);

l4=new JLabel("Gender");

l4.setBounds(50, 120, 100, 30);

C.add(l4);

b1=new JRadioButton("Male");

b1.setBounds(150, 120, 100, 30);

C.add(b1);

b2=new JRadioButton("Female");

b2.setBounds(250, 120, 100, 30);

C.add(b2);

b3=new JRadioButton("Others");

b3.setBounds(350, 120, 100, 30);

C.add(b3);

ButtonGroup bg=new ButtonGroup();

bg.add(b1);

bg.add(b2);

bg.add(b3);

l5=new JLabel("Date of Birth");

l5.setBounds(50, 150, 100, 30);

C.add(l5);

String day[]=new String[32];

day[0]="Day";

for(int i=1; i<=31; i++)

day[i]=Integer.toString(i);

cb1=new JComboBox(day);

cb1.setBounds(150, 150, 50, 30);

C.add(cb1);

String []month={"Month", "January","February", "March", "April", "May", "June", "July", "August", "Sptemeber", "October", "November", "December"};

cb2=new JComboBox(month);

cb2.setBounds(200, 150, 100, 30);

C.add(cb2);

int startYear = 1900;

int endYear = 2023;

int numberOfYears = endYear - startYear + 1;

String[] years = new String[numberOfYears +1];

years[0]="Year";

for (int i = 1; i <= numberOfYears; i++) {

years[i] = Integer.toString(startYear + i);

}

cb3=new JComboBox(years);

cb3.setBounds(300, 150, 100, 30);

C.add(cb3);

c1=new JCheckBox("I agree the given terms and conditions.");

c1.setBounds(50, 200, 400, 30);

C.add(c1);

c1.addActionListener(this);

submit=new JButton("Submit");

submit.setBounds(100, 250, 100, 30);

C.add(submit);

submit.addActionListener(this);

setVisible(true);

}

public void actionPerformed(ActionEvent e){

if(e.getSource()== submit){

if(c1.isSelected()){

String name=t1.getText();

String email=t2.getText();

String phone=t3.getText();

if(b1.isSelected())

gender ="Male";

else if(b2.isSelected())

gender ="Female";

else if(b3.isSelected())

gender ="Others";

selectedDay = (String) cb1.getSelectedItem();

selectedMth = (String) cb2.getSelectedItem();

selectedYear = (String) cb3.getSelectedItem();

try{

FileOutputStream fout=new FileOutputStream("registration.txt", true);

String s2="\n Name: "+name+", Email: "+email+ ", Phone number: "+ phone+ ", Gender: "+gender+ ", Dob: "+selectedDay+" "+selectedMth+" "+selectedYear;

byte b[]=s2.getBytes();

fout.write(b);

fout.close();

}catch(Exception ex){

System.out.println(ex);

}

JOptionPane.showMessageDialog(this,"Registration Successful ");

}

else

JOptionPane.showMessageDialog(this,"Select the terms and condition.");

}

}

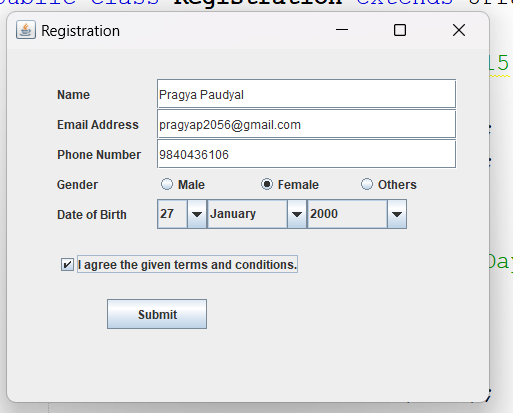
public static void main(String[] args) {

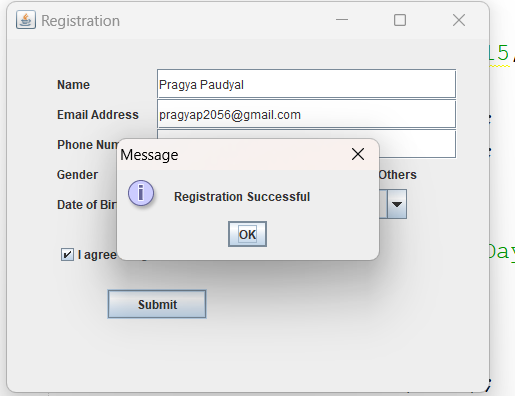
new Registration();

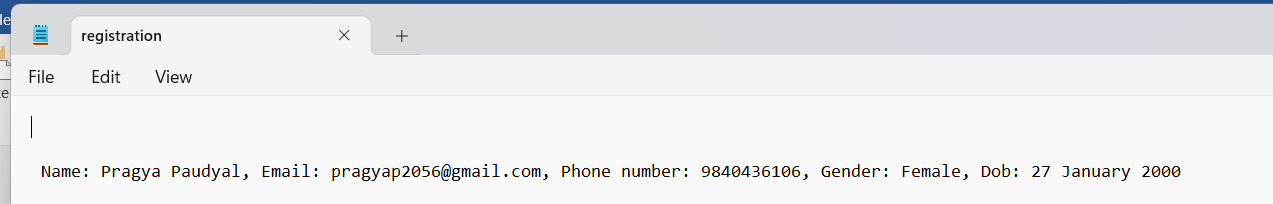
}

}

**Output:**







**CONCLUSION:**

In this lab we have successfully, created a registration form using the concept of swing, event handling and file handling.

# LAB 02: A java program to create a Login Form using the concept of Swing, Event Handling and File Handling.

**OBJECTIVE:** To create a Login Form using the concept of Swing, Event Handling and File Handling.

**THEORY:**

•Swing is a set of graphical user interface (GUI) components that are included in the Java Standard Edition (Java SE) platform. It provides a rich set of user interface components like buttons, text fields, labels, tables, and more, which can be used to create desktop applications.

File handling in Java is the process of reading from and writing to files using Java programming language. Java provides various classes and methods that can be used to handle files, such as FileReader, FileWriter, FileInputStream, FileOutputStream, BufferedReader, BufferedWriter, and more.

**Methods used:**

1. **add(Component c)** : adds component to container.
2. **addActionListenerListener(ActionListener d)** : add actionListener for specified component
3. **setBackground(Color c)** : sets the background color of the specified container
4. **setSize(int a, int b)** : sets the size of container to specified dimensions.
5. **setText(String s)** : sets the text of the label to s.
6. **getText()** : returns the text of the label.

**Source code:**

import java.awt.Color;

import java.awt.Container;

import java.awt.Cursor;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import javax.swing.\*;

public class Login extends JFrame implements ActionListener {

private Container c;

private JLabel l1,l2,l3,l4;

private JPasswordField p1;

private JTextField t1;

private JTextArea ta;

private JButton b1;

public Login()

{

setTitle("Frame");

setLocationRelativeTo(null);

setSize(700,700);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

c=this.getContentPane();

c.setBackground(Color.gray);

c.setLayout(null);

l1=new JLabel("Username");

l1.setBounds(50,100,100,40);

c.add(l1);

Font f1=new Font("Arial",Font.BOLD,20);

l1.setFont(f1);

t1=new JTextField();

t1.setBounds(150,100,300,40);

c.add(t1);

l2=new JLabel("Password");

l2.setBounds(50,180,100,40);

c.add(l2);

Font f2=new Font("Arial",Font.BOLD,20);

l2.setFont(f2);

p1=new JPasswordField();

p1.setBounds(150,180,300,40);

p1.setText("Pragya@123");

p1.setEchoChar((char)0);

c.add(p1);

l3=new JLabel("Address");

l3.setBounds(50,260,100,40);

c.add(l3);

Font f3=new Font("Arial",Font.BOLD,20);

l3.setFont(f3);

ta=new JTextArea();

ta.setBounds(150,250,300,40);

ta.setLineWrap(true);

c.add(ta);

b1=new JButton("Login");

b1.setBounds(200,300,100,40);

c.add(b1);

Cursor cr = new Cursor(Cursor.HAND\_CURSOR);

b1.setCursor(cr);

b1.addActionListener(this);

setVisible(true);

}

public void actionPerformed(ActionEvent ev){

if(ev.getSource()== b1){

try{

FileOutputStream f1 = new FileOutputStream ("Login.txt");

String a = "Username:"+t1.getText()+"Password:"+p1.getText()+"Address:"+ta.getText();

byte b1[]=a.getBytes();

f1.write(b1);

f1.close();

}

catch(IOException e){

System.out.println(e);

}

JOptionPane.showMessageDialog(null, "Login Information Saved Successfully!", "Success", JOptionPane.INFORMATION\_MESSAGE);

}

}

public static void main(String[] args)

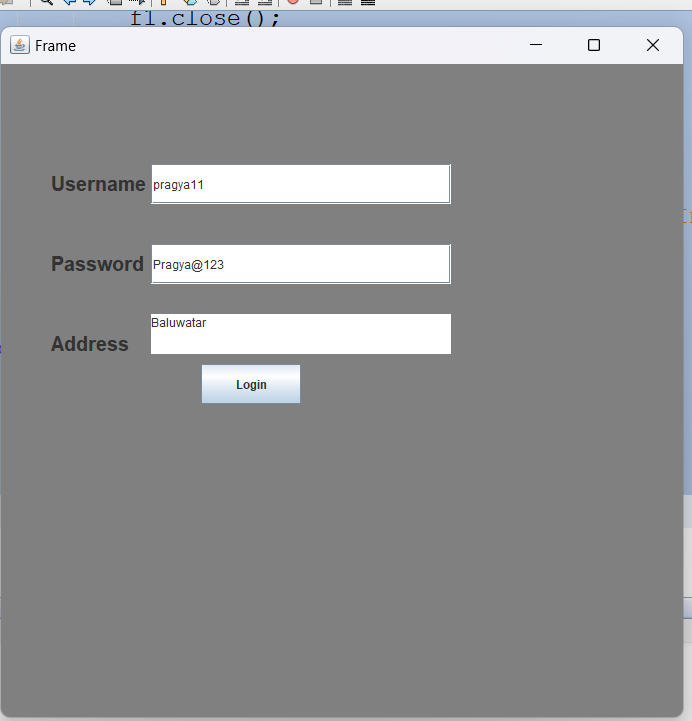
{

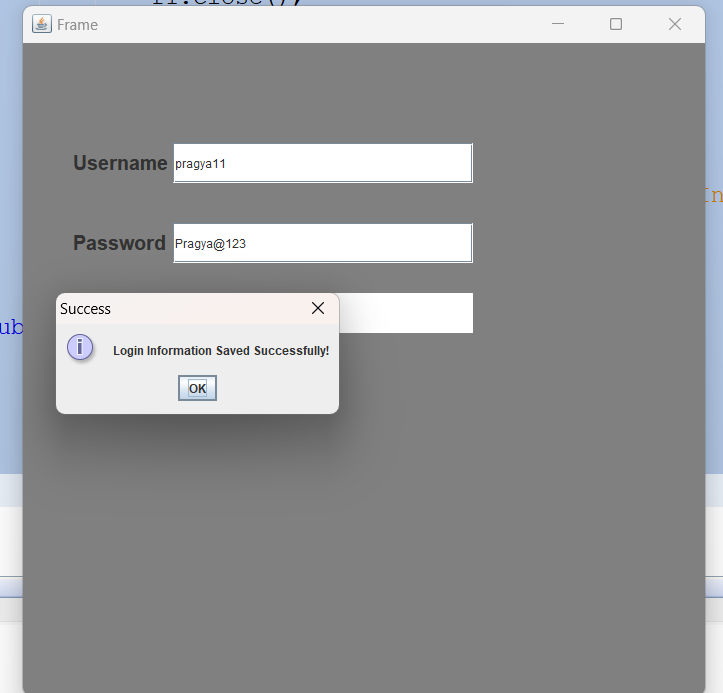
new Login();

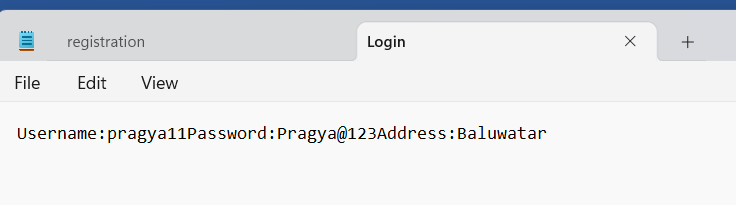
}

}

**OUTPUT:**

****

****

****

**CONCLUSION:**

In this lab we have successfully, created a login form using the concept of swing, event handling and file handling.

# LAB 03: A java program to add a content in a table using the concept of Swing and Event Handling.

**OBJECTIVE:** To add a content in a table using the concept of Swing and Event Handling.

**THEORY:**

Swing is a set of graphical user interface (GUI) components that are included in the Java Standard Edition (Java SE) platform. It provides a rich set of user interface components like buttons, text fields, labels, tables, and more, which can be used to create desktop applications.

**Methods used:**

1. **add(Component c)** : adds component to container.
2. **addActionListenerListener(ActionListener d)** : add actionListener for specified component
3. **setBackground(Color c)** : sets the background color of the specified container
4. **setSize(int a, int b)** : sets the size of container to specified dimensions.
5. **setText(String s)** : sets the text of the label to s.
6. **getText()** : returns the text of the label.

**Source code:**

import java.awt.Container;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

public class ProjTable extends JFrame {

private Container c;

private JTable tb;

private JTextField t1,t2,t3;

private JButton b1,b2,b3;

public ProjTable() {

setVisible(true);

setTitle("Jtable");

setSize(600,600);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

c=this.getContentPane();

Object data[][] = {{1,"Ram",12},{2,"Sita",13},{3,"Romeo",1},{5,"you",15}};

String [] column = {"RN","Name","Age"};

DefaultTableModel mode = new DefaultTableModel(data,column);

tb= new JTable(mode);

//add(new JScrollPane(tb));

setLayout(new GridLayout(3,1));

JPanel Panel1 = new JPanel();

add(new JScrollPane(tb));

add(new JPanel());

add(Panel1);

t1 = new JTextField();

t2 = new JTextField();

t3 = new JTextField();

b1= new JButton("Add");

b2= new JButton("Update");

b3= new JButton("Del");

Panel1.setLayout(new GridLayout(3,3));

Panel1.add(new JLabel("Roll no"));

Panel1.add(t1);

Panel1.add(b1);

tb.addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

int rowindex=tb.getSelectedRow();

int rollno = (int)mode.getValueAt(rowindex, 0);

String name = (String)mode.getValueAt(rowindex,1);

int age = (int)mode.getValueAt(rowindex, 2);

t1.setText(String.valueOf(rollno));

t2.setText(name);

t3.setText(String.valueOf(age));

}

});

b1.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {

if(t1.getText().toString().isEmpty()||t2.getText().toString().isEmpty()||t3.getText().toString().isEmpty())

{

JOptionPane.showMessageDialog(null,"please fill all Fields","Error",JOptionPane.ERROR\_MESSAGE);

}

else{

int rollno = Integer.parseInt(t1.getText().toString());

String name = t2.getText().toString();

int age = Integer.parseInt(t3.getText().toString());

Object[]newrow={rollno,name,age};

mode.addRow(newrow);

t1.setText(null);

t2.setText(null);

t3.setText(null);

}

}

});

b2.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {

if(t1.getText().toString().isEmpty()||t2.getText().toString().isEmpty()||t3.getText().toString().isEmpty())

{

JOptionPane.showMessageDialog(null,"please fill all Fields","Error",JOptionPane.ERROR\_MESSAGE);

}

else{

int rollno = Integer.parseInt(t1.getText().toString());

String name = t2.getText().toString();

int age = Integer.parseInt(t3.getText().toString());

int row = tb.getSelectedRow();

mode.setValueAt(rollno,row,0);

mode.setValueAt(name,row,1);

mode.setValueAt(age,row,2);

t1.setText(null);

t2.setText(null);

t3.setText(null);

}

}

});

b3.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e) {

if(tb.getSelectedRow()==-1)

{

JOptionPane.showMessageDialog(null,"please select a row","Message",JOptionPane.INFORMATION\_MESSAGE);

}

else{

int sel = JOptionPane.showConfirmDialog(null,"Are u sure?","Confirm",JOptionPane.YES\_NO\_OPTION);

if(sel==JOptionPane.YES\_NO\_OPTION)

{

mode.removeRow(tb.getSelectedRow());

t1.setText(null);

t2.setText(null);

t3.setText(null);

}}

}

});

tb.setSelectionMode(ListSelectionModel.SINGLE\_SELECTION);

Panel1.add(new JLabel("name"));

Panel1.add(t2);

Panel1.add(b2);

Panel1.add(new JLabel("age"));

Panel1.add(t3);

Panel1.add(b3);

validate();

}

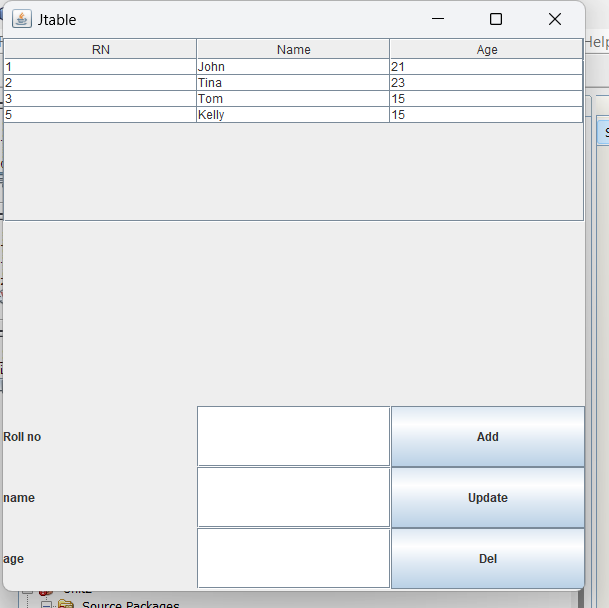
public static void main(String[] args) {

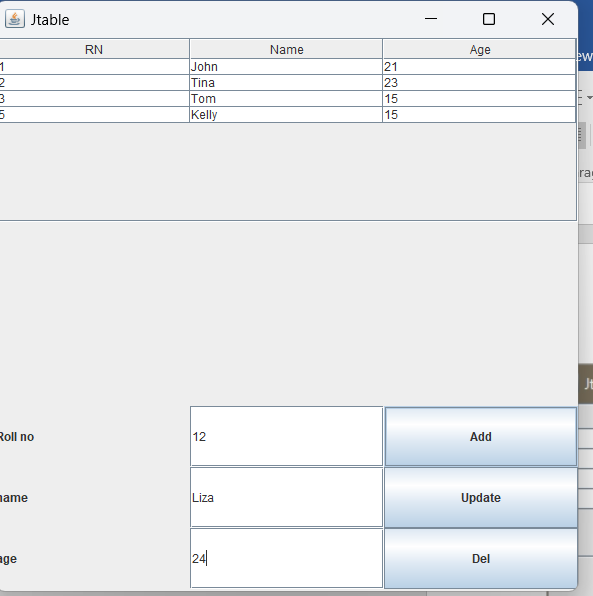
new ProjTable();

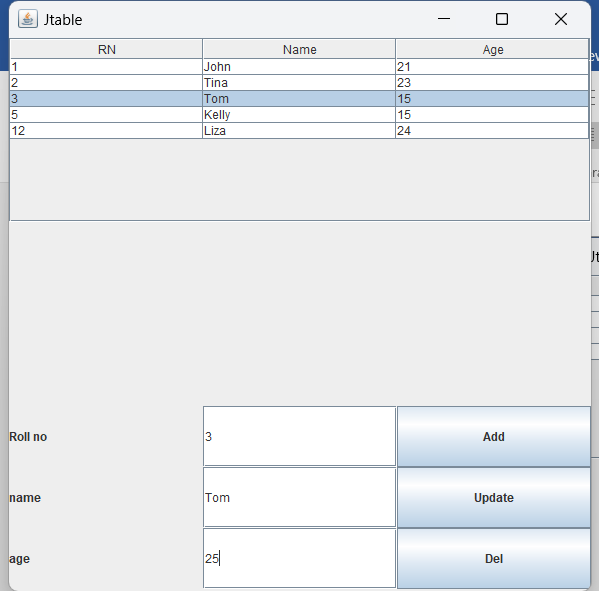
}

}

**OUTPUT:**



****

****

**CONCLUSION:**

In this lab session, we successfully add a content in a table using the concept of Swing and Event Handling.

# LAB 04: A program to create a simple calculator GUI using the concept of Swing, Event Handling in Java programming Language.

**OBJECTIVE:** To create a simple calculator GUI using the concept of Swing, Event Handling in Java programming Language.

**THEORY:**

Java Swing is a GUI (graphical user Interface) widget toolkit for Java. Java Swing is a part of Oracle’s Java foundation classes. Java Swing is an API for providing graphical user interface elements to Java Programs. Swing was created to provide more powerful and flexible components than Java AWT (Abstract Window Toolkit).  
**Methods used:**

1. **add(Component c)** : adds component to container.
2. **addActionListenerListener(ActionListener d)** : add actionListener for specified component
3. **setBackground(Color c)** : sets the background color of the specified container
4. **setSize(int a, int b)** : sets the size of container to specified dimensions.
5. **setText(String s)** : sets the text of the label to s.
6. **getText()** : returns the text of the label.

**SOURCE CODE:**

import java.awt.event.\*;

import javax.swing.\*;

import java.awt.\*;

class CalculatorDemo extends JFrame implements ActionListener {

static JFrame f;

static JTextField l;

String s0, s1, s2;

CalculatorDemo()

{

s0 = s1 = s2 = "";

}

public static void main(String args[])

{

f = new JFrame("calculator");

CalculatorDemo c = new CalculatorDemo();

l = new JTextField(16);

l.setEditable(false);

JButton b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, ba, bs, bd, bm, be, beq, beq1;

b0 = new JButton("0");

b1 = new JButton("1");

b2 = new JButton("2");

b3 = new JButton("3");

b4 = new JButton("4");

b5 = new JButton("5");

b6 = new JButton("6");

b7 = new JButton("7");

b8 = new JButton("8");

b9 = new JButton("9");

beq1 = new JButton("=");

ba = new JButton("+");

bs = new JButton("-");

bd = new JButton("/");

bm = new JButton("\*");

beq = new JButton("C");

be = new JButton(".");

JPanel p = new JPanel();

bm.addActionListener(c);

bd.addActionListener(c);

bs.addActionListener(c);

ba.addActionListener(c);

b9.addActionListener(c);

b8.addActionListener(c);

b7.addActionListener(c);

b6.addActionListener(c);

b5.addActionListener(c);

b4.addActionListener(c);

b3.addActionListener(c);

b2.addActionListener(c);

b1.addActionListener(c);

b0.addActionListener(c);

be.addActionListener(c);

beq.addActionListener(c);

beq1.addActionListener(c);

p.add(l);

p.add(ba);

p.add(b1);

p.add(b2);

p.add(b3);

p.add(bs);

p.add(b4);

p.add(b5);

p.add(b6);

p.add(bm);

p.add(b7);

p.add(b8);

p.add(b9);

p.add(bd);

p.add(be);

p.add(b0);

p.add(beq);

p.add(beq1);

f.add(p);

f.setSize(230, 240);

f.setLocationRelativeTo(null);

f.show();

}

public void actionPerformed(ActionEvent e)

{

String s = e.getActionCommand();

if ((s.charAt(0) >= '0' && s.charAt(0) <= '9') || s.charAt(0) == '.') {

if (!s1.equals(""))

s2 = s2 + s;

else

s0 = s0 + s;

l.setText(s0 + s1 + s2);

}

else if (s.charAt(0) == 'C') {

s0 = s1 = s2 = "";

l.setText(s0 + s1 + s2);

}

else if (s.charAt(0) == '=') {

double te;

if (s1.equals("+"))

te = (Double.parseDouble(s0) + Double.parseDouble(s2));

else if (s1.equals("-"))

te = (Double.parseDouble(s0) - Double.parseDouble(s2));

else if (s1.equals("/"))

te = (Double.parseDouble(s0) / Double.parseDouble(s2));

else

te = (Double.parseDouble(s0) \* Double.parseDouble(s2));

l.setText(s0 + s1 + s2 + "=" + te);

s0 = Double.toString(te);

s1 = s2 = "";

}

else {

if (s1.equals("") || s2.equals(""))

s1 = s;

else {

double te;

if (s1.equals("+"))

te = (Double.parseDouble(s0) + Double.parseDouble(s2));

else if (s1.equals("-"))

te = (Double.parseDouble(s0) - Double.parseDouble(s2));

else if (s1.equals("/"))

te = (Double.parseDouble(s0) / Double.parseDouble(s2));

else

te = (Double.parseDouble(s0) \* Double.parseDouble(s2));

s0 = Double.toString(te);

s1 = s;

s2 = "";

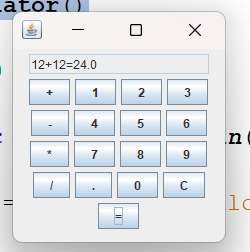
}

l.setText(s0 + s1 + s2);

}

}}

**OUTPUT:**



**CONCLUSION:**

In this lab of Advanced Java Programming, we successfully created a simple calculator GUI using the elements of Java Swing and the java event handling.

# LAB 05A: A program using TCP such that client sends number to server and displays its factorial. The server computes factorial of the number received from client.

**OBJECTIVE:** To implement a program using TCP such that client sends number to server and displays its factorial. The server computes factorial of the number received from client.

**THEORY:**

**Java Socket Programming:**

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:

1. IP Address of Server, and
2. Port number.

TCP/IP ServerSocket Class

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

Constructor

ServerSocket(int port):Creates a server socket, bound to the specified port.

Method

public Socket accept():returns the socket and establish a connection between server and client.

TCP/IP Client Sockets

The client in socket programming must know two information:

IP Address of Server

Port number.

Constructor:

Socket(InetAddress address,int port):creates stream socket and connects it to the specified port number at the specified IP address.

**SOURCE CODE:**

**Server.java**

import java.io.\*;

import java.net.\*;

class Server

{

public static void main(String args[])

{

try

{

ServerSocket ss=new ServerSocket(1064);

System.out.println("Waiting for Client Request");

Socket s=ss.accept();

BufferedReader br;

PrintStream ps;

String str;

br=new BufferedReader(new InputStreamReader(s.getInputStream()));

str=br.readLine();

System.out.println("Received number");

int x=Integer.parseInt(str);

int fact=1;

for(int i=1;i<=x;i++)

fact=fact\*i;

ps=new PrintStream(s.getOutputStream());

ps.println(String.valueOf(fact));

br.close();

ps.close();

s.close();

ss.close();

}

catch(Exception e)

{

System.out.println(e);

}

}}

**Client.java**

import java.io.\*;

import java.net.\*;

class Client

{

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

{

Socket s=new Socket(InetAddress.getLocalHost(),1064);

BufferedReader br;

PrintStream ps;

String str;

System.out.println("Enter a number :");

br=new BufferedReader(new InputStreamReader(System.in));

ps=new PrintStream(s.getOutputStream());

ps.println(br.readLine());

br=new BufferedReader(new InputStreamReader(s.getInputStream()));

str=br.readLine();

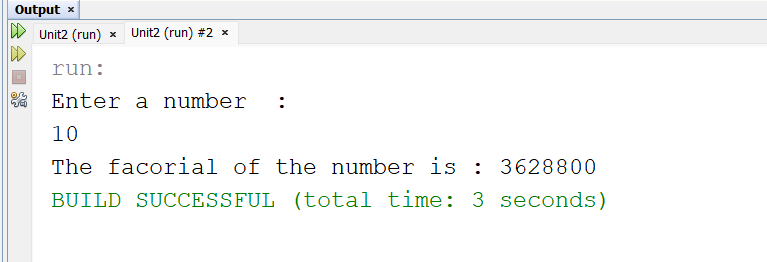
System.out.println("The facorial of the number is : "+str);

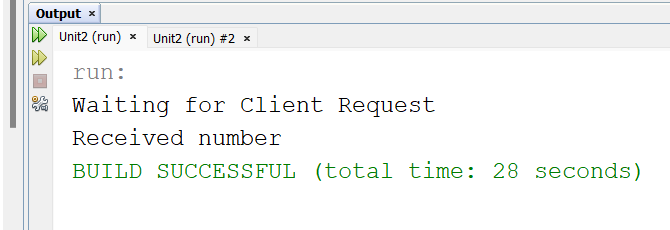
br.close();

ps.close();

}}

**OUTPUT:**





**CONCLUSION:**

In this lab session, we successfully implement the concepts of Network Programming: TCP/IP using the java.net package classes in Java programming language.

# LAB 05B: A java program using UDP showing that the sending and receiving of message using Datagram Packet and Datagram Socket class.

**OBJECTIVE:** To implement a java program using UDP that show the sending and receiving of message using Datagram Packet and Datagram Socket class.

**THEORY:**

**Java DatagramSocket and DatagramPacket**

Java DatagramSocket and DatagramPacket classes are used for connection-less socket programming using the UDP instead of TCP.

**Datagram**

Datagrams are collection of information sent from one device to another device via the established network. When the datagram is sent to the targeted device, there is no assurance that it will reach to the target device safely and completely. It may get damaged or lost in between. Likewise, the receiving device also never know if the datagram received is damaged or not. The UDP protocol is used to implement the datagrams in Java.

**Java DatagramSocket class**

Java DatagramSocket class represents a connection-less socket for sending and receiving datagram packets. It is a mechanism used for transmitting datagram packets over network. A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.

**SOURCE CODE:**

For **sender.java**

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

class Sender {

public static void main(String[] args)throws Exception {

DatagramSocket ds= new DatagramSocket();

String str= "Message sent by server";

InetAddress ip = InetAddress.getLocalHost();

DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(),ip, 6666);

ds.send(dp);

System.out.println("Message sent!");

ds.close();

}

}

For **receiver.java**

SOURCE CODE:

import java.net.DatagramPacket;

import java.net.DatagramSocket;

public class Receiver {

public static void main(String[] args) throws Exception {

DatagramSocket ds= new DatagramSocket(6666);

byte[] buf= new byte[1024];

DatagramPacket dp=new DatagramPacket(buf, 1024);

ds.receive(dp);

String str= new String(dp.getData(),0,dp.getLength());

System.out.println(str);

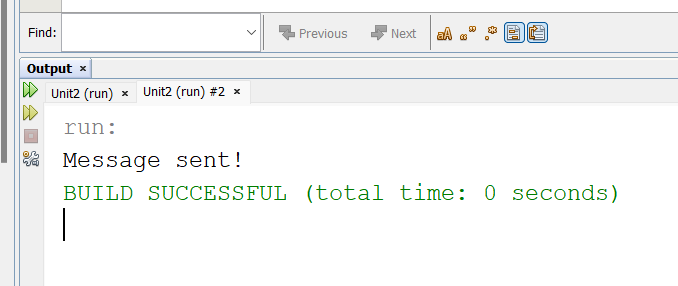
System.out.println("Message received!");

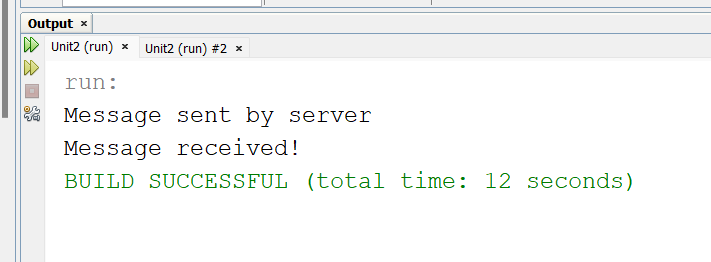
ds.close();

}

}

**OUTPUT:**





**CONCLUSION:**

In this lab session, we successfully implement the concepts of Network Programming: UDP using the java.net package classes in Java programming language.

# LAB 06: You are hired by a reputed software company which is going to design an application for "Movie Rental System". Your responsibility is to design a schema named MRS and create a table named Movie(id, Title, Genre, Language, Length). Write a program to design a GUI form to take input for this table and insert the data into table after clicking the OK button.

**OBJECTIVE:** To implement a program that design a GUI form that take input for table movie and insert the data into table movie after clicking OK button.

**THEORY:**

**JDBC:**

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

JDBC-ODBC Bridge Driver,

Native Driver,

Network Protocol Driver, and

Thin Driver

**SOURCE CODE:**

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

import java.sql.\*;

public class Movie extends JFrame implements ActionListener {

JFrame jf;

JTextField t1, t2,t3,t4;

JLabel l1,l2,l3,l4;

JButton b1;

public Movie(){

jf = new JFrame("Movie Rental System");

l1= new JLabel("Title");

l2= new JLabel("Genre");

l3= new JLabel("Language");

l4= new JLabel("Length");

t1= new JTextField(10);

t2= new JTextField(10);

t3= new JTextField(10);

t4= new JTextField(10);

b1= new JButton("ADD");

jf.add(l1);

jf.add(t1);

jf.add(l2);

jf.add(t2);

jf.add(l3);

jf.add(t3);

jf.add(l4);

jf.add(t4);

jf.add(b1);

jf.setSize(500,700);

jf.setLayout(new FlowLayout());

b1.addActionListener(this);

jf.setVisible(true);

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

}

public void actionPerformed(ActionEvent ae){

try{

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

Connection conn=

DriverManager.getConnection("jdbc:mysql://localhost:3306/mrs","root","");

String sql="insert into movie(title, genre, language, length) values (?,?,?,?)";

PreparedStatement ps= conn.prepareStatement(sql);

ps.setString(1, t1.getText());

ps.setString(2, t2.getText());

ps.setString(3, t3.getText());

ps.setString(4, t4.getText());

int i=ps.executeUpdate();

System.out.println("no. of rows updated ="+i);

ps.close();

conn.close();

// System.out.println("Inserted successfully");

}catch(Exception se){

System.out.println(se);

}

}

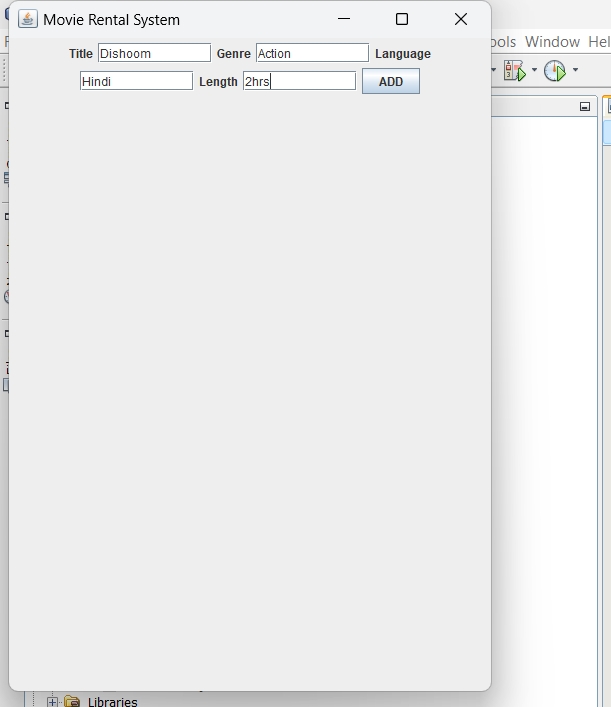
public static void main(String[] args) {

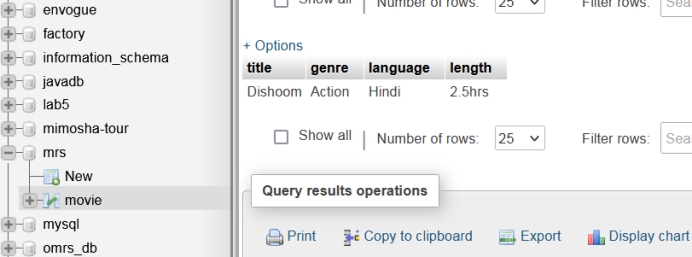
new Movie();

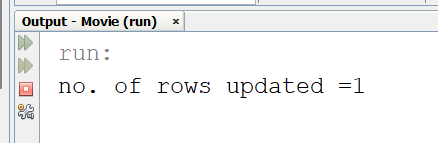
}

}

**Output:**







**Conclusion:**

In this lab session, we successfully connect to the MySQL database using JDBC connection using above Movie Rental System GUI.

# **LAB 07: A servlet application to print current date and time, communicate between html and servlet and to create an auto refreshed page**.

**OBJECTIVE:** To implement a servlet application to:

1. Print current date and time.
2. Communicate between Html and Servlet.
3. Create an auto refreshed page.

**THEORY:**

**Servlet:**

A servlet is a Java programming language class used to extend the capabilities of servers that host applications accessed via a request-response programming model

Three methods are central to the life cycle of a servlet

They are init(),service() and delete().

The javax.servletand javax.servlet.httppackages provide interfaces and classes for writing

Servlets.

## **7.1 To print current date and time:**

**SOURCE CODE:**

**Servlet.java**

package com.innovator;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

public class servlet extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out;

try {

out = response.getWriter();

out.println(" <html>");

out.println("<h1>The current date is: "+java.time.LocalDate.now()+"</h3>");

out.println("<h1>And, the current time is: "+java.time.LocalTime.now()+"</h3>");

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

} catch (IOException e) {

e.printStackTrace();

}

}

}

**datetime.html**

<!DOCTYPE html>

<html>

<head>

<title>TODO supply a title</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<a href="/ServeltExam/servlet"><button>Show Date and time</button></a>

</body>

</html>

**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>MyServlet</servlet-name>

<servlet-class>MyServlet</servlet-class>

</servlet>

<servlet>

<servlet-name>servlet</servlet-name>

<servlet-class>servlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>MyServlet</servlet-name>

<url-pattern>/MyServlet</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>servlet</servlet-name>

<url-pattern>/servlet</url-pattern>

</servlet-mapping>

<session-config>

<session-timeout>

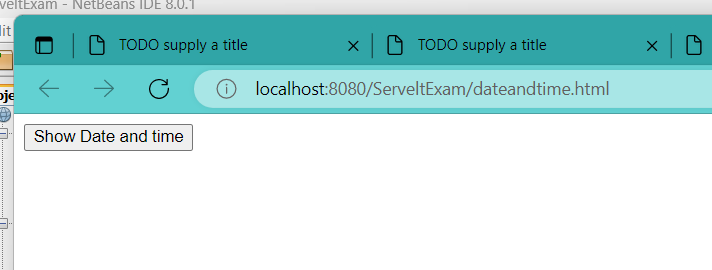
30

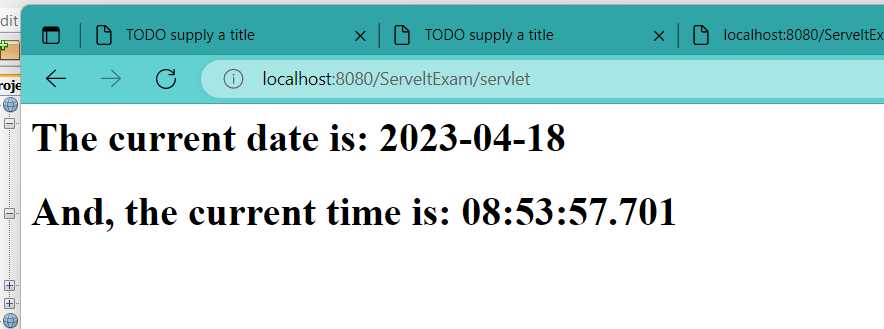
</session-timeout>

</session-config>

</web-app>

**OUTPUT:**





## **7.2 Communicate between Html and servlet.**

**SOURCE CODE:**

Index.html

<!DOCTYPE html>

<html>

<head>

<title>TODO supply a title</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

</head>

<body>

<form action="/ServeltExam/MyServlet" method="post">

<input type="text" name="fname" placeholder="enter full name">

<input type="radio" name="ge

nder" value="male">Male

<input type="radio" name="gender" value="female">Female

<select name="country">

<option value="Nepal">Nepal</option>

<option value="US">US</option>

<option value="India">India</option>

</select>

<input type="submit">

</form>

</body>

</html>

MyServlet.java

import java.io.\*;

import java.util.Enumeration;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class MyServlet extends HttpServlet {

String msg="";

PrintWriter out;

public void init() throws ServletException

{

msg="hello world: my first servlet program";

}

public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException,IOException

{

response.setContentType("text/html");

out=response.getWriter();

Enumeration<String> paramNames=request.getParameterNames();

while(paramNames.hasMoreElements())

{

String paramName=(String)paramNames.nextElement();

String value=request.getParameter(paramName);

out.println("<h1>"+paramName+":"+value+"</h1>");

}

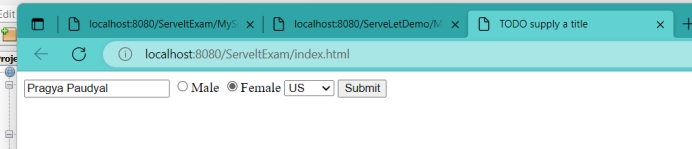
}

public void destroy()

{ out.close();

}

**Output:**





## **7.3 Create an auto refreshed page.**

**SOURCE CODE:**

Refresh.java file

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Calendar;

import java.util.GregorianCalendar;

public class Refresh extends HttpServlet {

public void doGet(HttpServletRequest request, HttpServletResponse response) {

response.setIntHeader("Refresh", 1);

// Set response content type

response.setContentType("text/html");

// Get current time

Calendar calendar = new GregorianCalendar();

String am\_pm;

int hour = calendar.get(Calendar.HOUR);

int minute = calendar.get(Calendar.MINUTE);

int second = calendar.get(Calendar.SECOND);

if(calendar.get(Calendar.AM\_PM) == 0)

am\_pm = "AM";

else

am\_pm = "PM";

String CT = hour+":"+ minute +":"+ second +" "+ am\_pm;

PrintWriter out;

try {

out = response.getWriter();

out.println(" <html>");

out.println("<h1 align='center'>Auto Refresh Page</h1><hr>");

out.println("<h3 align='center'>Current time: "+CT+"</h3>");

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

} catch (IOException e) {

e.printStackTrace();

}

}

}

Index.html

<html>

<head>

<title>Start page</title>

</head>

<body>

<h2>My servlet page!</h2>

<a href="/ServeltExam/Refresh"><button>Refresh Page</button></a>

</body>

</html>

Web.xml file

<?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>MyServlet</servlet-name>

<servlet-class>MyServlet</servlet-class>

</servlet>

<servlet>

<servlet-name>servlet</servlet-name>

<servlet-class>servlet</servlet-class>

</servlet>

<servlet>

<servlet-name>Refresh</servlet-name>

<servlet-class>Refresh</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>MyServlet</servlet-name>

<url-pattern>/MyServlet</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>servlet</servlet-name>

<url-pattern>/servlet</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>Refresh</servlet-name>

<url-pattern>/Refresh</url-pattern>

</servlet-mapping>

<session-config>

<session-timeout>

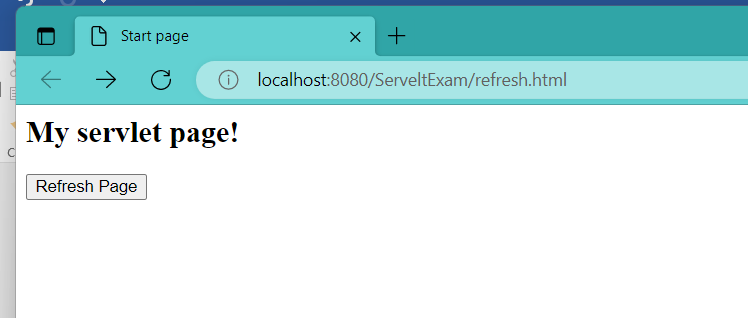
30

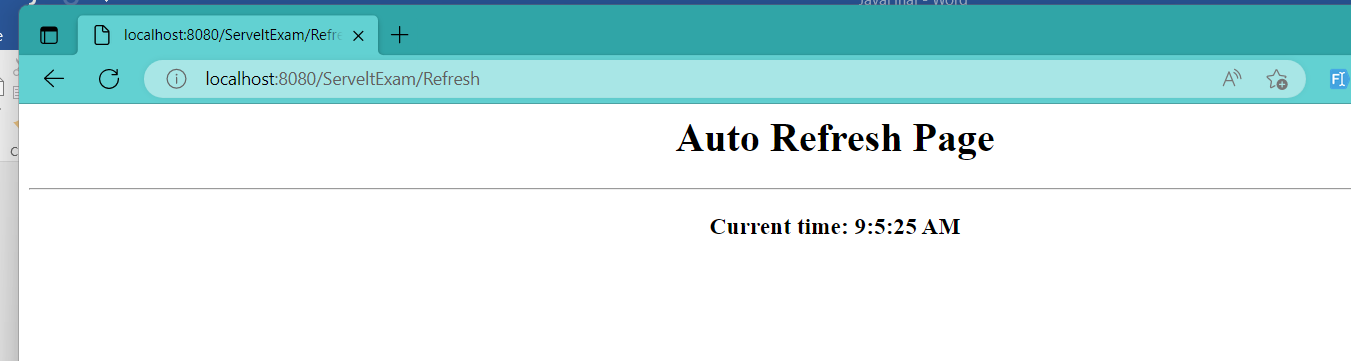
</session-timeout>

</session-config>

</web-app>

**OUTPUT:**





**CONCLUSION:**

In this lab of Advanced Java Programming, we successfully implemented the concept of servlet using Maven and Glash Fish Server in Netbeans.

# LAB 08: A JSP application program to login.

**OBJECTIVE:** To create a login page using jsp.

**THEORY:**

**JSP:**

Java Server Pages (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.

**SOURCE CODE:**

index.jsp file

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Login Page</title>

</head>

<body>

<h1>Login</h1>

<form method="post" action="login.jsp">

<label for="username">Username:</label>

<input type="text" id="username" name="username" required><br><br>

<label for="password">Password:</label>

<input type="password" id="password" name="password" required><br><br>

<input type="submit" value="Login">

</form>

</body>

</html>

Login.jsp file

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Login Result</title>

</head>

<body>

<%

String username = request.getParameter("username");

String password = request.getParameter("password");

if(username.equals("admin") && password.equals("password")) {

RequestDispatcher dispatcher = request.getRequestDispatcher("success.jsp");

dispatcher.forward(request, response);

} else {

RequestDispatcher dispatcher = request.getRequestDispatcher("failed.jsp");

dispatcher.forward(request, response);

}

%>

</body>

</html>

Success.jsp file

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

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

<title>JSP Page</title>

</head>

<body>

<h1>Login Successful</h1>

</body>

</html>

Failed.jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

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

<title>JSP Page</title>

</head>

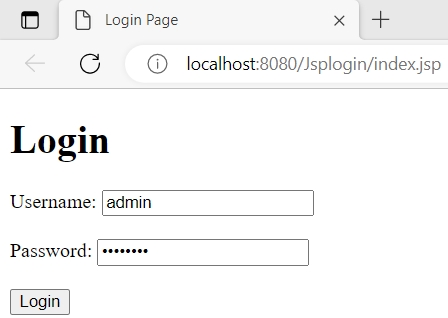
<body>

<h1>Login Failed</h1>

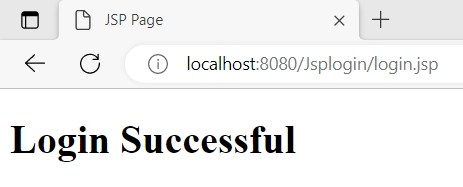
</body>

</html>

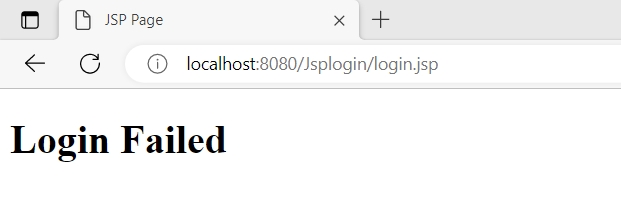
**OUTPUT:**



Login Sucessful



Login Failed



**CONCLUSION:**

In this lab of Advanced Java Programming, we successfully implemented the concept of JSP using Glass Fish Server in netbeans.

# LAB 09: RMI program to add two number.

**OBJECTIVE:** To implement an RMI that add two number.

**THEORY:**

**RMI:**

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.

**SOURCE CODE:**

**Adder.java interface**

import java.rmi.\*;

public interface Adder extends Remote {

public int add(int x,int y)throws RemoteException;

}

**Client.java**

import java.rmi.\*;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.NotBoundException;

import java.util.Scanner;

public class Client {

public static void main(String[] args) throws RemoteException{

Client c=new Client();

c.connectRemote();

}

public void connectRemote() throws RemoteException

{

try

{

Scanner sc=new Scanner(System.in);

Registry reg=LocateRegistry.getRegistry("localhost",9999);

Adder ad=(Adder)reg.lookup("Hello Server");

System.out.println("Enter two number");

int a=sc.nextInt();

int b=sc.nextInt();

System.out.println("Addition is"+ad.add(a,b));

}

catch(Exception e)

{

System.out.println(e);

}

}

}

**Server.java**

import java.rmi.\*;

import java.rmi.registry.\*;

import java.rmi.server.UnicastRemoteObject;

public class Server extends UnicastRemoteObject implements Adder{

public Server() throws RemoteException

{

super();

}

public int add(int x,int y)

{

return x+y;

}

public static void main(String[] args) throws RemoteException{

try{

Registry reg=LocateRegistry.createRegistry(9999);

reg.rebind("Hello Server", new Server());

System.out.println("Servlet is ready");

}

catch(Exception e)

{

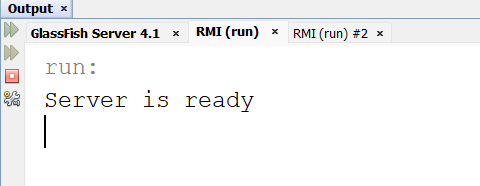
System.out.println(e);

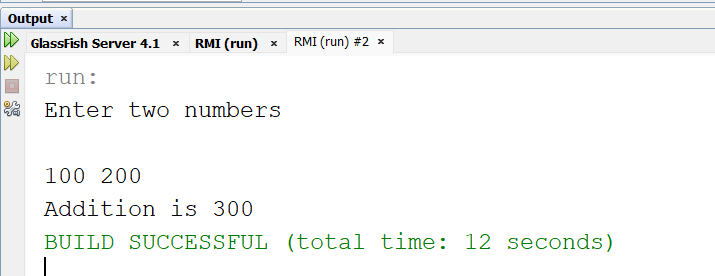
}

}

}

**OUTPUT:**

****



**CONCLUSION:**

In this lab of Advanced Java Programming, we successfully implemented the concept of Remote Method Invocation (RMI) using Java programming language.