INDEX

S.No.	TITLE	Sign
1.	Create a webpage that prints your name to the screen, print your name in Tahoma font, print a definition list with 5 items, Create links to five different pages, etc.	
2.	Program to demonstrate Swing components.	
3.	Configure Apache Tomcat and write a hello world JSP page.	
4.	Write a java program that connects to a database using JDBC and does add, delete and retrieve operations.	
5.	Create and develop a web application using JSF.	
6.	Write a program to implement a Java Beans to set and get values.	
7.	Create a Java application to demonstrate Socket Programming in Java.	
8.	Write a program to retrieve hostname--using methods in Inetaddress class.	
9.	Write a client-server program which displays the server machine's date and time on the client machine.	
10.	Create a table in the database containing the columns to store book details like: book name, authors, description, price and URL of the book's cover image. Using JSP and JDBC retrieve the details in the table and display them on the webpage.	
11.	Write a program to create a login page using Java Beans. Also validate the username and password from the database.	

Program 1: Create a webpage that prints your name to the screen, print your name in Tahoma font, print a definition list with 5 items, Create links to five different pages, etc.

CODE:

```
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
        body {
            font-family: Tahoma, sans-serif;
        }
</style>
</head>
<body>
<h1>xyz</h1>
<h2>Definition List</h2>
<dl class="definition-list">
<dt>Instagram</dt>
<dd>this is instagram</dd>
<dt>facebook</dt>
<dd>this is facebook</dd>
<dt>snapchat</dt>
<dd>this is snapchat</dd>
<dt>youtube</dt>
<dd>this is youtube</dd>
<dt>amazon</dt>
<dd>this is amazon</dd>
```

```
</dl>
<ul>
<li><a href="https://www.instagram.com/" target="_blank">instagram</a></li>
<li><a href="https://www.facebook.com/" target="_blank">facebook</a></li>
<li><a href="https://www.snapchat.com/" target="_blank">snapchat</a></li>
<li><a href="https://www.youtube.com/" target="_blank">youtube</a></li>
<li><a href="https://www.amazon.com/" target="_blank">amazon</a></li>
</ul>
</body>
</html>
```

OUTPUT:

xyz

Definition List

Instagram
    this is instagram
facebook
    this is facebook
snapchat
    this is snapchat
youtube
    this is youtube
amazon
    this is amazon

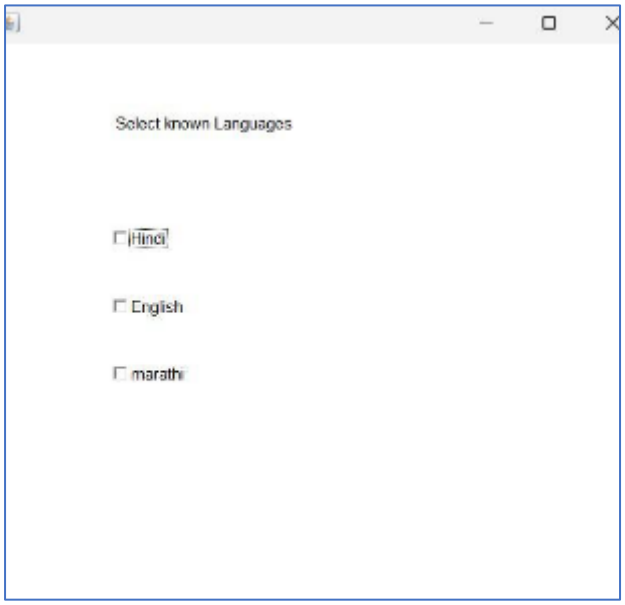
- instagram
- facebook
- snapchat
- youtube
- amazon

Program 2:Program to demonstrate Swing components.

CODE:

```
import java.awt.*;

class comp {

comp()
   {
      // Frame Created
      Frame f = new Frame();

      Label l1 = new Label("Select known Languages");

      l1.setBounds(100, 50, 120, 80);
    f.add(l1);

      // CheckBox created
      Checkbox c2 = new Checkbox("Hindi");
      c2.setBounds(100, 150, 50, 50);
    f.add(c2);

      // CheckBox created
      Checkbox c3 = new Checkbox("English");
      c3.setBounds(100, 200, 80, 50);
     f.add(c3);

      // CheckBox created
      Checkbox c4 = new Checkbox("marathi");
      c4.setBounds(100, 250, 80, 50);
     f.add(c4);

f.setSize(500, 500);
f.setLayout(null);
f.setVisible(true);
   }

   public static void main(String ar[]) { new comp(); }
}
```

OUTPUT:



Program 3:Configure Apache Tomcat and write a hello world JSP page.

CODE:

Step 1: Download and Install Apache Tomcat

1. Download the latest version of Apache Tomcat from the Apache Tomcat website.
2. Unzip the downloaded file to your desired directory.
3. Set the CATALINA_HOME environment variable to point to your Tomcat directory.

Step 2: Start Apache Tomcat

1. Open a terminal or command prompt.
2. Navigate to the bin directory inside your Tomcat installation.
3. Run the appropriate startup script:
    - For Windows: startup.bat
    - For macOS/Linux: ./startup.sh

You should see a message indicating that Tomcat has started successfully. By default, Tomcat runs on port 8080. To check if it's running, open a browser and go to:http://localhost:8080

Step 3: Create a JSP File

1. Go to the webapps directory inside your Tomcat installation.
2. Create a new directory named hello (this will be your application).
3. Inside the hello directory, create another directory named WEB-INF.
4. Inside the hello directory, create a new JSP file named index.jsp.

Step 4: Write Code for the "Hello World" JSP Page

Open **index.jsp** and add the following code:

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

<!DOCTYPE html>

<html>

<head>

<title>Hello World JSP</title>

</head>

<body>
```

```
<h1>Hello, World!</h1>

</body>

</html>
```

Step 5: Access the JSP Page

1. Make sure Tomcat is running.
2. In your browser, navigate to:

http://localhost:8080/hello/index.jsp

You should see the "Hello, World!" message displayed on the page.

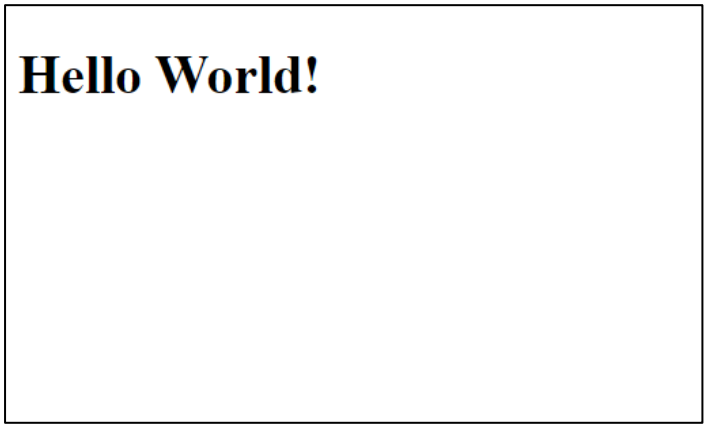
Step 6: Stop Apache Tomcat (Optional)

To stop Tomcat, go back to the bin directory and run the shutdown command:

- For Windows: shutdown.bat
- For macOS/Linux: ./shutdown.sh

And that’s it! You’ve successfully configured Apache Tomcat and created a basic JSP "Hello World" page.

OUTPUT:



Program 4:Write a java program that connects to a database using JDBC and does add, delete and retrieve operations.

CODE:

```
import java.sql.*;
public class jdbc1 {
   public static void main(String args[]) {
      String url = "jdbc:oracle:thin:@localhost:1521:xe";
      String username = "system";
      String pass = "12345";
      try {
Class.forName("oracle.jdbc.driver.OracleDriver");
         Connection con = DriverManager.getConnection(url, username, pass);
System.out.println("connection created");

//table creation
         Statement st = con.createStatement();
st.executeUpdate("create table trail(id integer,name varchar(10))");
System.out.println("table created successfully");

         String query = "Insert into trail(id,name) values(?,?)";

PreparedStatement p = con.prepareStatement(query);

p.setInt(1, 1);
p.setString(2, "abc");
p.executeUpdate();
p.setInt(1, 2);
p.setString(2, "def");
```

```
p.executeUpdate();
p.setInt(1, 3);
p.setString(2, "ghi");
p.executeUpdate();

//          fetching the data
PreparedStatementps = con.prepareStatement("Select * from trail");
ResultSetrs = ps.executeQuery();
        while (rs.next()) {
System.out.println("-----------------------------------");
System.out.println("id: " + rs.getInt("id"));
System.out.println("name: " + rs.getString("name"));


        }
    //deleting
        String del = "DELETE FROM trail where id=?";
PreparedStatementpstmt = con.prepareStatement(del);
pstmt.setInt(1, 3);
pstmt.executeUpdate();
System.out.println("record deleted successfully");
```

OUTPUT:

```
run:
connection created
table created successfully

 id: 1
 name: abc
 ------------------------------------
 id: 2
 name: def
 ------------------------------------
 id: 3
 name: ghi
 record deleted successfully

------------------------------------
id: 1
name: abc
------------------------------------
id: 2
name: def
BUILD SUCCESSFUL (total time: 1 second)
```

Program 5: Create and Develop a web application using JSF.

CODE:-

index.xhtml

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:h="http://xmlns.jcp.org/jsf/html"
     xmlns:f="http://xmlns.jcp.org/jsf/core">
   <h:head>
      <title>Student Registration form</title>
   </h:head>
   <h:body>
      <h:form>
         First Name : <h:inputText id="firstName" value
="#{student.firstName}" />
         <br></br><br/>
         Last Name : <h:inputText id ="lastName" value =
"#{student.lastName}" />
         <br/><br/>
         Country :
         <h:selectOneMenu value="#{student.country}">
            <f:selectItem itemValue="India" itemLabel="India"/>
            <f:selectItem itemValue="Brazil" itemLabel="Brazil"/>
            <f:selectItem itemValue="USA" itemLabel="USA"/>
         </h:selectOneMenu>
         <h:commandButton value="submit"  action="response"/>
      </h:form>
   </h:body>
</html>
```

response.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
```

```
and open the template in the editor.
-->
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h = "http://xmlns.jcp.org/jsf/html">
    <head>
        <title>TODO supply a title</title>
        <meta name="viewport" content="width=device-width, initial-
scale=1.0"/>
    </head>
    <body>
        Student name is : #{student.firstName} #{student.lastName}
        <br/>
        Country : #{student.country}
    </body>
</html>

package aa;

import javax.faces.bean.ManagedBean ;

@ManagedBean
public class student {
    private String firstName ;
    private String lastName ;
    private String country ;

//    create no-arg constructor

    public student(){

    }

//    define getter and setter method

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
```

```
        this.country = country;
    }


    public String getFirstName() {
        return firstName;
    }
public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

}
```

OUTPUT:

Program 6: Write a program to implement a Java Beans to set and get values.

CODE:

Index.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="bean.jsp" method="post">
        name:<input type="text" name="name"><br>
<input type="submit" value="submit"><br>
</form>
</body>
</html>
```

Bean.jsp:

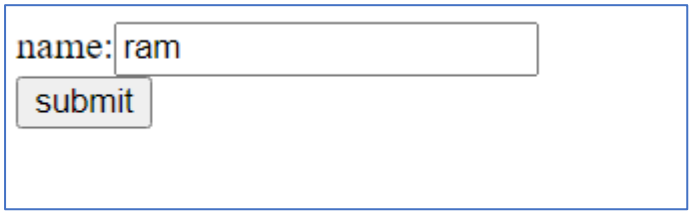
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body>
<jsp:useBean id="ob" class="aa.Bean" scope="application"></jsp:useBean>
<jsp:setProperty property="*" name="ob"></jsp:setProperty>
<jsp:getProperty name="ob" property="name"></jsp:getProperty><br>

</body>
</html>
```

Bean.java:

```
package aa;

import java.io.Serializable;
public class Bean implements Serializable{
```

```
    String name;

    public String getName() {
        return name;
    }

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

}
```

OUTPUT:



Program 7: Create a Java application to demonstrate Socket Programming in Java.

CODE:

sever.java:

```
package aa;

import java.io.DataInputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class server {
   public static void main(String agrs[]){
try{
            //create a server socket
ServerSocketsk = new ServerSocket(8854);

            //waiting for client to connect
System.out.println("Waiting for client to connect");
            Socket s =sk.accept();
System.out.println("client connected"+s.getInetAddress());

DataInputStream dis = new DataInputStream(s.getInputStream());
            String msg=dis.readUTF();
System.out.println(msg  );
         }
catch(Exception e){
System.out.println(e);
         }

   }
}
```

client.java:

```
package aa;

import java.io.DataOutputStream;
import java.net.Socket;

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

```
System.out.println("Client");
          //create a socket
         Socket s = new Socket("localhost", 8854);

DataOutputStream dos = new DataOutputStream(s.getOutputStream());
dos.writeUTF("hello by client to server");
      }
catch(Exception e){
System.out.println("Error");
   }

   }
}
```

OUTPUT:

```
run:
Waiting for client to connect
```

```
run:
Client
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
run:
Waiting for client to connect
client connected/127.0.0.1
hello by client to server
BUILD SUCCESSFUL (total time: 49 seconds)
```

Program 8: Write a program to retrieve hostname--using methods in Inetaddress class.

CODE:

```
package aa;

import java.net.InetAddress;

public class program{

   public static void main(String[] args) throws Exception {
System.out.println("LocalHost address");
InetAddress a1 = InetAddress.getLocalHost();
System.out.println(a1);

InetAddressad[] = InetAddress.getAllByName("www.youtube.com");
System.out.println("getAllByName");
      for (InetAddressa : ad) {
System.out.println(a);
      }
System.out.println("getByName");
InetAddress aa = InetAddress.getByName("www.youtube.com");
System.out.println(aa);
      byte b1[] = aa.getAddress();

   }
}
```

OUTPUT:

```
LocalHost address
PC05LAB3Dell3000/192.168.10.165
getAllByName
www.youtube.com/142.250.194.110
www.youtube.com/142.250.194.174
www.youtube.com/142.250.194.78
www.youtube.com/142.250.206.174
www.youtube.com/142.250.193.78
www.youtube.com/142.250.193.238
www.youtube.com/142.250.194.206
www.youtube.com/142.250.206.142
www.youtube.com/142.250.207.206
www.youtube.com/142.250.206.110
www.youtube.com/142.250.195.14
www.youtube.com/142.250.193.206
www.youtube.com/142.250.194.14
www.youtube.com/142.250.194.46
www.youtube.com/142.250.194.238
www.youtube.com/142.250.194.142
getByName
www.youtube.com/142.250.194.110
BUILD SUCCESSFUL (total time: 0 seconds)
```

Program 9: Write a client-server program which displays the server machine's date and time on the client machine.

CODE:

server.java:

```
package aa;

import java.io.IOException;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.text.SimpleDateFormat;
import java.util.Date;

public class server {
  public static void main(String[] args) {
      int port = 12345;

      try (ServerSocketserverSocket = new ServerSocket(port)) {
System.out.println("Server is waiting for a connection...");

          Socket clientSocket = serverSocket.accept();
System.out.println("Client connected");

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);

          String dateTime = getCurrentDateTime();

out.println(dateTime);

clientSocket.close();
System.out.println("Data sent to client and connection closed.");
      } catch (IOException e) {
System.out.println("Server Error: " + e.getMessage());
      }
  }

  private static String getCurrentDateTime() {
SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
      Date date = new Date();
      return formatter.format(date);
```

```
    }
}
```

client:

```
package aa;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;

public class client {
    public static void main(String[] args) {
        String serverAddress = "localhost";
        int port = 12345;

        try (Socket socket = new Socket(serverAddress, port)) {

BufferedReader input = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            String serverDateTime = input.readLine();
System.out.println("Server's Date and Time: " + serverDateTime);
        } catch (IOException e) {
System.out.println("Client Error: " + e.getMessage());
        }
    }
}
```

OUTPUT:

```
run:
Server is waiting for a connection...
Client connected
Data sent to client and connection closed.
BUILD SUCCESSFUL (total time: 11 seconds)
```

```
run:
Server's Date and Time: 2024-11-09 22:01:16
BUILD SUCCESSFUL (total time: 0 seconds)
```

Program 10: - Create a table in the database containing the columns to store book details like: book name, authors, description, price and URL of the book's cover image. Using JSP and JDBC retrieve the details in the table and display them on the webpage.

CODE:-

Index.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>
    <div>Book detail</div>
    <form action="books.jsp" method="post">
      <input type="submit" value="book detail"/>
    </form>


  </body>
</html>
```

Books.jsp:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1" %>
<!DOCTYPE html>
<html>
<head>
  <title>Books List</title>
</head>
```

```
<body>
    <h1>List of Books</h1>

    <!-- Define the database connection -->
    <sql:setDataSource var="dataSource" driver="oracle.jdbc.driver.OracleDriver"
                    url="jdbc:oracle:thin:@localhost:1521:xe"
                    user="system" password="12345"/>

    <!-- Execute the SQL query -->
    <sql:query var="booksQuery" dataSource="${dataSource}">
        SELECT * FROM Books
    </sql:query>

    <!-- Display the retrieved books in a table -->
    <table border="1">
        <tr>
            <th>Book Name</th>
            <th>Author</th>
            <th>Description</th>
            <th>Price</th>
            <th>Cover Image</th>
        </tr>

        <c:forEach var="book" items="${booksQuery.rows}">
            <tr>
                <td>${book.book_name}</td>
                <td>${book.author}</td>
                <td>${book.description}</td>
                <td>${book.price}</td>
```

```
            <td><img src="${book.cover_image_url}" alt="Cover Image" width="100"
height="150"></td>
          </tr>
      </c:forEach>
   </table>
</body>
</html>
```

OUTPUT:-

List of Books

Book Name	Author	Description	Price	Cover Image
The Great Gatsby	F. Scott Fitzgerald	The Great Gatsby by F. Scott Fitzgerald is a 1925 novel about the tragic love story of Jay Gatsby, a self-made millionaire, and his pursuit of Daisy Buchanan, a wealthy young woman. The story is told from the perspective of Nick Carraway, a neighbor of Gatsby's who becomes involved in Gatsby's failed love affair.	3	
To Kill a Mockingbird	Harper Lee	Set in small-town Alabama, the novel is a bildungsroman, or coming-of-age story, and chronicles the childhood of Scout and Jem Finch as their father Atticus defends a Black man falsely accused of rape	30	

Program 11: Write a program to create a login page using Java Beans. Also validate the username and password from the database.

CODE:-

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Login</title>
</head>
<body>
   <h2>Login</h2>
   <form action="login.jsp" method="POST">
      <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:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
```

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<jsp:useBean id="ob" class="aa.loginbean" scope="application" />
<jsp:setProperty property="*" name="ob" />

<html>
<head>
   <title>Login Validation</title>
</head>
<body>
   <h2>Login Validation</h2>

   <!-- Set up database connection -->
   <sql:setDataSource var="dataSource" driver="oracle.jdbc.driver.OracleDriver"
                  url="jdbc:oracle:thin:@localhost:1521:xe"
                  user="system" password="12345"/>

   <!-- Validate login credentials from database -->
   <sql:query var="checkLogin" dataSource="${dataSource}">
      SELECT * FROM user_detail WHERE username = '${ob.username}' AND password =
'${ob.password}'
   </sql:query>

   <!-- Use JSTL to handle conditional display -->
   <c:choose>
      <c:when test="${not empty checkLogin.rows}">
         <!-- If credentials match, show welcome message -->
         <h3>Welcome, ${ob.username}!</h3>
      </c:when>
      <c:otherwise>
```

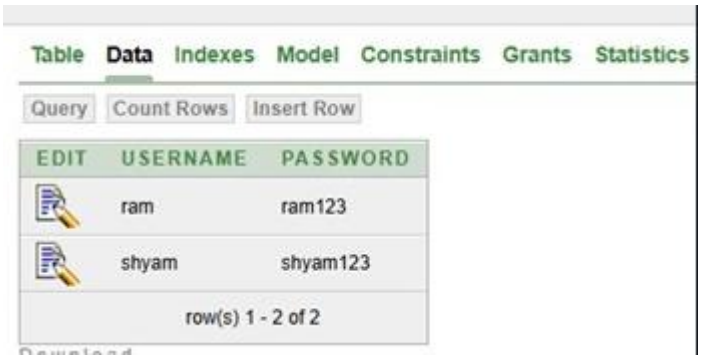
```
        <!-- If credentials do not match, show error -->
        <h3 style="color:red;">Invalid Username or Password!</h3>
      </c:otherwise>
    </c:choose>
</body>
</html>
```

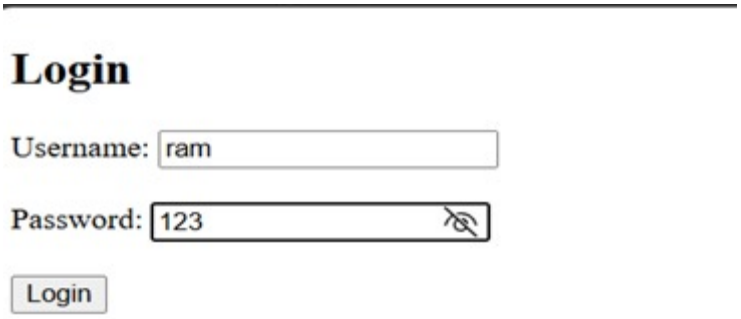
Loginbean.jsp:

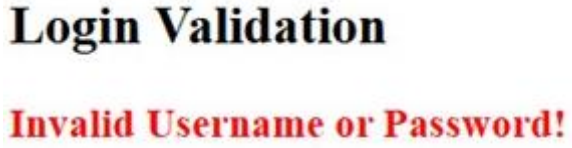
```
package aa;

public class loginbean {
    private String username;
    private String password;

    // Getters and Setters
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
```

```
        this.password = password;
    }
}
```

OUTPUT:







Login

Username: ram

Password: ram123

Login

Login Validation

Welcome, ram!