**Advanced JAVA Programming**

**Index**

|  |  |
| --- | --- |
| **Exp no** | **Title of Experiments** |
| **Group A (ALL)** | |
| 1. | Write a program to demonstrate status of key on an Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown. |
| 2. | Write a program to create a frame using AWT. Implement mouseClicked, mouseEntered() and mouseExited() events. Frame should become visible when the mouse enters it. |
| 3. | Develop a GUI which accepts the information regarding the marks for all the subjects of a student in the examination. Display the result for a student in a separate window |
| 4. | Write a program to insert and retrieve the data from the database using JDBC |
| 5. | Develop an RMI application which accepts a string or a number and checks that string or number is palindrome or not. |
| 6. | Write a program to demonstrate the use of InetAddress class and its factory methods. |
| **Group B (Any Two)** | |
| 7. | A. Write Servlet (procedure for client side) to display the username and password accepted from the client.  B. Write Servlet (procedure for server side) to display the username and password accepted from the client. |
| 8. | Write program with suitable example to develop your remote interface, implement your RMI server, implement application that create your server, also develop security policy file. |
| **Group C (Any Two)** | |
| 9. | Write a simple JSP page to display a simple message (It may be a simple html page). |
| 10. | Create a simple calculator application using servlet |
| 11. | Create a registration servlet in Java using JDBC. Accept the details such as Username, Password, Email, and Country from the user using HTML Form and store the registration details in the database. |

**Exp1: Write a program to demonstrate status of key on an Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown.**

//code

import java.applet.Applet;

import java.awt.Graphics;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

/\*

<applet code="KeyEventDemo" width=400 height=200>

</applet>

\*/

public class KeyEventDemo extends Applet implements KeyListener {

String message = "";

int x = 20, y = 60;

public void init() {

addKeyListener(this); // Register KeyListener

requestFocus(); // Applet should get keyboard focus

}

public void keyPressed(KeyEvent ke) {

message = "Key Pressed: " + ke.getKeyChar();

repaint();

}

public void keyReleased(KeyEvent ke) {

message = "Key Released: " + ke.getKeyChar();

repaint();

}

public void keyTyped(KeyEvent ke) {

message = "Key Typed: " + ke.getKeyChar();

repaint();

}

public void paint(Graphics g) {

g.drawString(message, x, y);

}

}

**How to Compile and Run**

**Step 1: Save as KeyEventDemo.java**

**Step 2: Compile**

javac KeyEventDemo.java

**Step 3: Run using Applet Viewer**

appletviewer KeyEventDemo.java

**Exp 2: Write a program to create a frame using AWT. Implement mouseClicked, mouseEntered() and mouseExited() events. Frame should become visible when the mouse enters it.**

//code

import java.awt.\*;

import java.awt.event.\*;

public class MouseEventDemoAWT extends Frame implements MouseListener {

String message = "";

public MouseEventDemoAWT() {

// Set title and size

setTitle("Mouse Event Demo");

setSize(400, 300);

// Add mouse listener to the frame

addMouseListener(this);

// Set layout and background color

setLayout(new FlowLayout());

setBackground(Color.LIGHT\_GRAY);

// Make frame invisible initially

setVisible(false);

}

// Called when mouse is clicked

public void mouseClicked(MouseEvent me) {

message = "Mouse Clicked at (" + me.getX() + ", " + me.getY() + ")";

repaint();

}

// Called when mouse enters the frame

public void mouseEntered(MouseEvent me) {

message = "Mouse Entered Frame";

setVisible(true); // Make frame visible

repaint();

}

// Called when mouse exits the frame

public void mouseExited(MouseEvent me) {

message = "Mouse Exited Frame";

repaint();

}

public void mousePressed(MouseEvent me) {}

public void mouseReleased(MouseEvent me) {}

// Paint method to display messages

public void paint(Graphics g) {

g.drawString(message, 100, 150);

}

// Main method to run the program

public static void main(String[] args) {

MouseEventDemoAWT frame = new MouseEventDemoAWT();

// Initially not visible; will be shown when mouse enters it

}

}

**Compile and Run code**

**Step 1: Save the program as MouseEventDemoAWT.java**

**Step 2: Compile**

javac MouseEventDemoAWT.java

**Step 3: Run**

java MouseEventDemoAWT

**Exp 3: Develop a GUI which accepts the information regarding the marks for all the subjects of a student in the examination. Display the result for a student in a separate window.**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class StudentMarksGUI extends JFrame implements ActionListener {

JTextField[] marksFields = new JTextField[5];

JButton resultButton;

public StudentMarksGUI() {

setTitle("Student Marks Entry");

setSize(400, 300);

setLayout(new GridLayout(7, 2, 10, 10));

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

for (int i = 0; i < 5; i++) {

add(new JLabel("Subject " + (i + 1) + " Marks:"));

marksFields[i] = new JTextField();

add(marksFields[i]);

}

resultButton = new JButton("Show Result");

resultButton.addActionListener(this);

add(resultButton);

setLocationRelativeTo(null); // Center on screen

setVisible(true);

}

public void actionPerformed(ActionEvent e) {

try {

int total = 0;

for (JTextField tf : marksFields) {

int mark = Integer.parseInt(tf.getText());

if (mark < 0 || mark > 100) {

throw new NumberFormatException("Invalid mark range");

}

total += mark;

}

int average = total / 5;

String grade;

if (average >= 90) grade = "A+";

else if (average >= 80) grade = "A";

else if (average >= 70) grade = "B";

else if (average >= 60) grade = "C";

else if (average >= 50) grade = "D";

else grade = "Fail";

showResultWindow(total, average, grade);

} catch (NumberFormatException ex) {

JOptionPane.showMessageDialog(this, "Please enter valid integer marks between 0 and 100.");

}

}

private void showResultWindow(int total, int average, String grade) {

JFrame resultFrame = new JFrame("Student Result");

resultFrame.setSize(300, 200);

resultFrame.setLayout(new GridLayout(4, 1));

resultFrame.add(new JLabel("Total Marks: " + total));

resultFrame.add(new JLabel("Average Marks: " + average));

resultFrame.add(new JLabel("Grade: " + grade));

JButton closeButton = new JButton("Close");

closeButton.addActionListener(e -> resultFrame.dispose());

resultFrame.add(closeButton);

resultFrame.setLocationRelativeTo(this);

resultFrame.setVisible(true);

}

public static void main(String[] args) {

new StudentMarksGUI();

}

}

**How to Compile and Run**

**Step 1: Save as StudentMarksGUI.java**

**Step 2: Compile**

javac StudentMarksGUI.java

**Step 3: Run**

java StudentMarksGUI

**Exp 4 : Write a program to insert and retrieve the data from the database using JDBC.**

### 1. ****Create Database and Table in MySQL****

CREATE DATABASE StudentDB;

USE StudentDB;

CREATE TABLE students (

rno INT PRIMARY KEY,

name VARCHAR(100)

);

**2. StudentJDBC.java**

import java.sql.\*;

import java.util.Scanner;

public class StudentJDBC {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Connection con = null;

PreparedStatement insertStmt = null;

Statement selectStmt = null;

ResultSet rs = null;

String url = "jdbc:mysql://localhost:3306/StudentDB"; // Change port/db if needed

String user = "root"; // Replace with your DB username

String password = "root"; // Replace with your DB password

try {

// Load MySQL JDBC Driver

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

// Establish connection

con = DriverManager.getConnection(url, user, password);

System.out.println("Connected to the database.");

// Insert student

System.out.print("Enter Roll No: ");

int rno = sc.nextInt();

sc.nextLine(); // consume newline

System.out.print("Enter Name: ");

String name = sc.nextLine();

String insertSQL = "INSERT INTO students (rno, name) VALUES (?, ?)";

insertStmt = con.prepareStatement(insertSQL);

insertStmt.setInt(1, rno);

insertStmt.setString(2, name);

insertStmt.executeUpdate();

System.out.println("Record inserted successfully.");

// Retrieve and display all students

String selectSQL = "SELECT \* FROM students";

selectStmt = con.createStatement();

rs = selectStmt.executeQuery(selectSQL);

System.out.println("\nStudent Records:");

while (rs.next()) {

System.out.println("Roll No: " + rs.getInt("rno") + ", Name: " + rs.getString("name"));

}

} catch (Exception e) {

e.printStackTrace();

} finally {

try { if (rs != null) rs.close(); } catch (Exception e) {}

try { if (insertStmt != null) insertStmt.close(); } catch (Exception e) {}

try { if (selectStmt != null) selectStmt.close(); } catch (Exception e) {}

try { if (con != null) con.close(); } catch (Exception e) {}

}

}

}

**Compile and Run**

#### Compile

javac -cp .;mysql-connector-java-8.x.x.jar StudentJDBC.java

#### Run

java -cp .;mysql-connector-java-8.x.x.jar StudentJDBC

**Replace mysql-connector-java-8.x.x.jar with your actual MySQL JDBC jar file path.**

**Exp 5 : Develop an RMI application which accepts a string or a number and checks that string or number is palindrome or not.**

Now here create all java file and put in same directory

Let us consider folder name RMIExample put all java and policy file in same directory

//code

**1.PalindromeInterface.java**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface PalindromeInterface extends Remote {

boolean checkPalindrome(String input) throws RemoteException;

}

**2. PalindromeServer.java**

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

public class PalindromeServer extends UnicastRemoteObject implements PalindromeInterface {

public PalindromeServer() throws RemoteException {

super();

}

@Override

public boolean checkPalindrome(String input) throws RemoteException {

// Remove spaces and convert to lower case for case-insensitive comparison

String cleanedInput = input.replaceAll("\\s+", "").toLowerCase();

String reversed = new StringBuilder(cleanedInput).reverse().toString();

// Check if input is equal to its reverse

return cleanedInput.equals(reversed);

}

public static void main(String[] args) {

try {

PalindromeServer server = new PalindromeServer();

// Create and bind the RMI registry

java.rmi.registry.LocateRegistry.createRegistry(1099);

java.rmi.Naming.rebind("PalindromeService", server);

System.out.println("Palindrome RMI Server is running...");

} catch (Exception e) {

e.printStackTrace();

}

}

}

**3. PalindromeClient.java**

import java.rmi.Naming;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.Scanner;

public class PalindromeClient {

public static void main(String[] args) {

try {

// Get the RMI registry

Registry registry = LocateRegistry.getRegistry("localhost", 1099);

// Lookup the remote object

PalindromeInterface stub = (PalindromeInterface) registry.lookup("PalindromeService");

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string or number to check for palindrome: ");

String input = scanner.nextLine();

// Call the remote method

boolean result = stub.checkPalindrome(input);

if (result) {

System.out.println("\"" + input + "\" is a palindrome.");

} else {

System.out.println("\"" + input + "\" is not a palindrome.");

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

**policy.policy (Security Policy) (create text file and named it policy.policy and put following code)**

**grant {**

**permission java.security.AllPermission;**

**};**

## How to Compile and Run the Application

### Step 1: Compile all files

**javac PalindromeInterface.java PalindromeServer.java PalindromeClient.java**

**Step 2: Start the RMI Registry**

Make sure the RMI registry is running:

**start rmiregistry**

**Step 3: Start the RMI Server**

Run the RMI server:

**java -Djava.security.policy=policy.policy PalindromeServer**

**Step 4: Run the RMI Client**

Now, run the client to test the palindrome functionalit

**java -Djava.security.policy=policy.policy PalindromeClient**

**Exp 6 :Write a program to demonstrate the use of InetAddress class and its factory methods.**

InetAddressExample.java

import java.net.\*;

import java.io.\*;

public class InetAddressExample {

public static void main(String[] args) {

try {

// 1. Get InetAddress for a given hostname (e.g., google.com)

InetAddress googleAddress = InetAddress.getByName("google.com");

System.out.println("Google's IP Address: " + googleAddress.getHostAddress());

// 2. Get InetAddress for the local host (your own machine)

InetAddress localAddress = InetAddress.getLocalHost();

System.out.println("Local Hostname: " + localAddress.getHostName());

System.out.println("Local Host IP Address: " + localAddress.getHostAddress());

// 3. Get InetAddress from an IP address (example using byte array for 8.8.8.8)

byte[] ip = {(byte) 8, (byte) 8, (byte) 8, (byte) 8}; // IP for 8.8.8.8

InetAddress addressFromIP = InetAddress.getByAddress(ip);

System.out.println("IP Address from byte array: " + addressFromIP.getHostAddress());

// 4. Checking if a host is reachable

boolean isReachable = googleAddress.isReachable(2000); // Timeout 2 seconds

System.out.println("Is google.com reachable? " + isReachable);

// 5. Display all IP addresses for the local host (hostname resolution)

InetAddress[] localAddresses = InetAddress.getAllByName(localAddress.getHostName());

System.out.println("All IP addresses for local host:");

for (InetAddress addr : localAddresses) {

System.out.println(addr.getHostAddress());

}

} catch (UnknownHostException e) {

System.err.println("Host could not be resolved: " + e.getMessage());

} catch (IOException e) {

System.err.println("Error while checking reachability: " + e.getMessage());

}

}

}

### Step 1: Save the program as InetAddressExample.java

### Step 2: Compile the Program

javac InetAddressExample.java

### Step 3: Run the Program

java InetAddressExample

**Explanation of methods**

1. **InetAddress.getByName("hostname")**: Resolves the IP address of a given hostname.
2. **InetAddress.getLocalHost()**: Returns the InetAddress for the local machine.
3. **InetAddress.getByAddress(byte[] addr)**: Creates an InetAddress object from an IP address represented by a byte array.
4. **InetAddress.isReachable(int timeout)**: Checks if the host is reachable within the specified timeout in milliseconds.
5. **InetAddress.getAllByName(String host)**: Retrieves all IP addresses associated with a given host.

**Group B**

**Exp 7A:Write Servlet (procedure for client side) to display the username and password accepted from the client.**

**File 1: Login.html**

<!DOCTYPE html>

<html>

<head>

<title>Login Form</title>

</head>

<body>

<h2>Login Form</h2>

<form action="LoginServlet" method="post">

Username: <input type="text" name="username" /><br><br>

Password: <input type="password" name="password" /><br><br>

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

</form>

</body>

</html>

**File 2: LoginServlet.java**

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("/LoginServlet")

public class LoginServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Set content type

response.setContentType("text/html");

// Get writer

PrintWriter out = response.getWriter();

// Read parameters from request

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

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

// Display the data

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

out.println("<h2>Login Details Received</h2>");

out.println("<p><strong>Username:</strong> " + username + "</p>");

out.println("<p><strong>Password:</strong> " + password + "</p>");

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

}

}

**Exp 7 B. Write Servlet (procedure for server side) to display the username and password accepted from the client.**

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;

public class LoginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

// Handles POST requests

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Set response content type

response.setContentType("text/html");

// Get output stream to write response

PrintWriter out = response.getWriter();

// Get form parameters

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

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

// Display the received data

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

out.println("<h2>Login Details Received on Server</h2>");

out.println("<p><strong>Username:</strong> " + username + "</p>");

out.println("<p><strong>Password:</strong> " + password + "</p>");

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

}

}

**Exp 8: Write program with suitable example to develop your remote interface, implement your RMI server, implement application that create your server, also develop security policy file.**

**Steps 1:**

**Create folder with name RMIExample and put all java code here**.

**MyRemoteInterface.java**

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface MyRemoteInterface extends Remote {

String sayHello(String name) throws RemoteException;

int add(int a, int b) throws RemoteException;

int subtract(int a, int b) throws RemoteException;

}

**MyRemoteImpl.java**

import java.rmi.server.UnicastRemoteObject;

import java.rmi.RemoteException;

public class MyRemoteImpl extends UnicastRemoteObject implements MyRemoteInterface {

public MyRemoteImpl() throws RemoteException {

super();

}

@Override

public String sayHello(String name) throws RemoteException {

return "Hello, " + name + "! Welcome to Java RMI.";

}

@Override

public int add(int a, int b) throws RemoteException {

return a + b;

}

@Override

public int subtract(int a, int b) throws RemoteException {

return a - b;

}

}

**RMIServer.java**

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class RMIServer {

public static void main(String[] args) {

try {

MyRemoteInterface stub = new MyRemoteImpl();

// Start the RMI registry

Registry registry = LocateRegistry.createRegistry(1099);

// Bind the object

registry.rebind("HelloService", stub);

System.out.println("RMI Server is running...");

} catch (Exception e) {

e.printStackTrace();

}

}

}

**RMIClient.java**

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

public class RMIClient {

public static void main(String[] args) {

try {

Registry registry = LocateRegistry.getRegistry("localhost", 1099);

MyRemoteInterface stub = (MyRemoteInterface) registry.lookup("HelloService");

// Call sayHello

String greeting = stub.sayHello("Alice");

System.out.println("Greeting: " + greeting);

// Call add

int sum = stub.add(10, 20);

System.out.println("Addition (10 + 20): " + sum);

// Call subtract

int diff = stub.subtract(50, 15);

System.out.println("Subtraction (50 - 15): " + diff);

} catch (Exception e) {

e.printStackTrace();

}

}

}

**policy.policy**

**grant {**

**permission java.security.AllPermission;**

**};**

**Execution:**

**Step1: Compile all java files**

**Ex. Javac filename.java**

**Step2:** Start RMI Registry

**Open terminal and run this comande ---------------> start rmiregistry**

Step3: start server

java -Djava.security.policy=policy.policy RMIServer

Step 4: Run Client code

java -Djava.security.policy=policy.policy RMIClient

**Group C**

**Exp 9: Write a simple JSP page to display a simple message (It may be a simple html page).**

**1.simpleMessage.jsp**

<!DOCTYPE html>

<html>

<head>

<title>Simple JSP Page</title>

</head>

<body>

<h2>Welcome to the Simple JSP Page!</h2>

<p>This is a basic JSP page displaying a simple message.</p>

</body>

</html>

### Steps to Run the JSP Page:

1. **Create the JSP File**: Save the above code in a file named simpleMessage.jsp.
2. **Deploy the File**:
   * Place the simpleMessage.jsp file inside the **webapp** directory of your servlet container (e.g., Apache Tomcat).
3. **Start Your Servlet Container**:
   * If you're using **Apache Tomcat**, you can start it by running startup.bat (Windows) or startup.sh (Linux/Mac).
4. **Access the JSP Page in a Web Browser**:
5. Open your browser and go to:

http://localhost:8080/your-web-app-name/simpleMessage.jsp

**Exp 10: Create a simple calculator application using servlet.**

**Calculator Application using Servlets** that performs basic arithmetic operations (addition, subtraction, multiplication, and division). The application consists of two main components:

1. **HTML Form** where the user inputs two numbers and selects an operation.
2. **Servlet** that processes the user's input and performs the arithmetic operation.

**1. CalculatorServlet.java**

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

public class CalculatorServlet extends HttpServlet {

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// Get input numbers from the form

double num1 = Double.parseDouble(request.getParameter("num1"));

double num2 = Double.parseDouble(request.getParameter("num2"));

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

double result = 0;

String message = "";

// Perform calculation based on the selected operation

switch (operation) {

case "add":

result = num1 + num2;

message = "Result: " + result;

break;

case "subtract":

result = num1 - num2;

message = "Result: " + result;

break;

case "multiply":

result = num1 \* num2;

message = "Result: " + result;

break;

case "divide":

if (num2 != 0) {

result = num1 / num2;

message = "Result: " + result;

} else {

message = "Error: Cannot divide by zero";

}

break;

default:

message = "Invalid operation";

break;

}

// Set the result message as a request attribute

request.setAttribute("result", message);

// Forward the request to the result.jsp page to display the result

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

dispatcher.forward(request, response);

}

}

**2. calculator.html**

<!DOCTYPE html>

<html>

<head>

<title>Simple Calculator</title>

</head>

<body>

<h2>Simple Calculator</h2>

<form action="CalculatorServlet" method="post">

<label for="num1">Enter first number:</label>

<input type="text" id="num1" name="num1" required><br><br>

<label for="num2">Enter second number:</label>

<input type="text" id="num2" name="num2" required><br><br>

<label for="operation">Choose operation:</label>

<select id="operation" name="operation" required>

<option value="add">Addition</option>

<option value="subtract">Subtraction</option>

<option value="multiply">Multiplication</option>

<option value="divide">Division</option>

</select><br><br>

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

</form>

</body>

</html>

**3. result.jsp** (JSP Page to Display Result)

<!DOCTYPE html>

<html>

<head>

<title>Calculation Result</title>

</head>

<body>

<h2>Calculation Result</h2>

<p>${requestScope.result}</p>

<br>

<a href="calculator.html">Go Back to Calculator</a>

</body>

</html>

**4. web.xml (Servlet Configuration)**

The web.xml file maps the servlet to a specific URL pattern.

<web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_1.xsd" version="3.1">

<servlet>

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

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

</servlet>

<servlet-mapping>

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

<url-pattern>/CalculatorServlet</url-pattern>

</servlet-mapping>

</web-app>

### Steps to Compile and Run the Application:

1. **Create the Servlet and JSP Files**:
   * Save the CalculatorServlet.java, calculator.html, and result.jsp in the correct locations in your web application structure (e.g., WEB-INF/classes for the servlet and webapp/ for HTML/JSP).
2. **Compile the Servlet**:
   * Compile the CalculatorServlet.java file:

javac -classpath "path\_to\_your\_servlet\_api.jar" CalculatorServlet.java

1. **Deploy the Application**:
   * Deploy the application in a servlet container like **Apache Tomcat**. Ensure that the compiled servlet (CalculatorServlet.class) is placed inside the WEB-INF/classes/ directory of your web app.
2. **Start Your Servlet Container**:
   * Start **Tomcat** by running startup.bat (Windows) or startup.sh (Linux/Mac).
3. **Access the Application**:
   * Open your web browser and go to:

http://localhost:8080/your-web-app-name/calculator.html

**Exp 11. Create a registration servlet in Java using JDBC. Accept the details such as Username, Password, Email, and Country from the user using HTML Form and store the registration details in the database.**

**Database Setup:**

First, create a database table to store the user registration details. For example, use the following SQL to create a table in your database:

CREATE TABLE user\_registration (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(50) NOT NULL,

password VARCHAR(100) NOT NULL,

email VARCHAR(100) NOT NULL,

country VARCHAR(50) NOT NULL

);

**JDBC Configuration:**

You will need the **JDBC Driver** for your database (e.g., MySQL, PostgreSQL). Ensure the driver is in your classpath, and you have a proper database connection URL.

**1. HTML Form for Registration (register.html):**

The user will enter their details using this form.

<!DOCTYPE html>

<html>

<head>

<title>User Registration</title>

</head>

<body>

<h2>User Registration Form</h2>

<form action="RegisterServlet" 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>

<label for="email">Email:</label>

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

<label for="country">Country:</label>

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

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

</form>

</body>

</html>

**2.JDBC Registration Servlet (RegisterServlet.java):**

This servlet processes the registration details and stores them in the database

import java.io.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.sql.\*;

public class RegisterServlet extends HttpServlet {

// JDBC details

private static final String DB\_URL = "jdbc:mysql://localhost:3306/your\_database\_name";

private static final String DB\_USERNAME = "your\_db\_username";

private static final String DB\_PASSWORD = "your\_db\_password";

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

// Retrieve form data from request

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

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

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

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

// JDBC connection and statement objects

Connection connection = null;

PreparedStatement preparedStatement = null;

try {

// Load JDBC driver

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

// Establish connection to the database

connection = DriverManager.getConnection(DB\_URL, DB\_USERNAME, DB\_PASSWORD);

// SQL query to insert the data into the database

String sql = "INSERT INTO user\_registration (username, password, email, country) VALUES (?, ?, ?, ?)";

// Create prepared statement to avoid SQL injection

preparedStatement = connection.prepareStatement(sql);

preparedStatement.setString(1, username);

preparedStatement.setString(2, password);

preparedStatement.setString(3, email);

preparedStatement.setString(4, country);

// Execute the update

int result = preparedStatement.executeUpdate();

// Check if the registration was successful

if (result > 0) {

response.getWriter().println("<h2>Registration Successful</h2>");

} else {

response.getWriter().println("<h2>Registration Failed</h2>");

}

} catch (Exception e) {

e.printStackTrace();

response.getWriter().println("<h2>Error: " + e.getMessage() + "</h2>");

} finally {

// Close resources

try {

if (preparedStatement != null) preparedStatement.close();

if (connection != null) connection.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

}

}

**3. web.xml Configuration (web.xml):**

Map the RegisterServlet to a URL pattern.

<web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_1.xsd" version="3.1">

<servlet>

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

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

</servlet>

<servlet-mapping>

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

<url-pattern>/RegisterServlet</url-pattern>

</servlet-mapping>

</web-app>

### ****Steps to Run the Application****:

1. **JDBC Driver Setup**:
   * Ensure you have the **JDBC driver** for your database (e.g., MySQL) in your project classpath.
2. **Create Database Table**:
   * Run the SQL query to create the user\_registration table in your database.
3. **Compile the Servlet**:
   * Compile RegisterServlet.java into the WEB-INF/classes directory.

javac -classpath "path\_to\_your\_servlet\_api.jar" RegisterServlet.java

1. **Deploy to Servlet Container**:
   * Deploy the application to a servlet container (e.g., **Apache Tomcat**). Place RegisterServlet.class in the WEB-INF/classes folder and register.html in the root of the web application.
2. **Start Tomcat Server**:
   * Start your **Tomcat** server by running startup.bat (Windows) or startup.sh (Linux/Mac).
3. **Access the Registration Form**:
   * Open your browser and go to:

http://localhost:8080/your-web-app-name/register.html

1. **Submit the Form**:
   * Fill in the form with user details and submit. The servlet will process the form, store the details in the database, and display a success message.