1. Write a Java program to illustrate multilevel inheritance (BoxBoxWeight-BoxShipment)

class Box {

private double width;

private double height;

private double depth;

// construct clone of an object

Box(Box ob) {

width = ob.width;

height = ob.height;

depth = ob.depth;

}

Box(double w, double h, double d) {

width = w;

height = h;

depth = d;

}

Box() {

width = -1; // use -1 to indicate

height = -1; // an uninitialized

depth = -1; // box

}

Box(double len) {

width = height = depth = len;

}

double volume() {

return width \* height \* depth;

}

}

// Add weight.

class BoxWeight extends Box {

double weight;

BoxWeight(BoxWeight ob) {

super(ob);

weight = ob.weight;

}

BoxWeight(double w, double h, double d, double m) {

super(w, h, d);

weight = m;

}

BoxWeight() {

super();

weight = -1;

}

BoxWeight(double len, double m) {

super(len);

weight = m;

}

}

class Shipment extends BoxWeight {

double cost;

Shipment(Shipment ob) {

super(ob);

cost = ob.cost;

}

Shipment(double w, double h, double d,

double m, double c) {

super(w, h, d, m);

cost = c;

}

Shipment() {

super();

cost = -1;

}

Shipment(double len, double m, double c) {

super(len, m);

cost = c;

}

}

class DemoShipment {

public static void main(String args[]) {

Shipment shipment1 =

new Shipment(10, 20, 15, 10, 3.41);

Shipment shipment2 =

new Shipment(2, 3, 4, 0.76, 1.28);

double vol;

vol = shipment1.volume();

System.out.println("Volume of shipment1 is " + vol);

System.out.println("Weight of shipment1 is "

+ shipment1.weight);

System.out.println("Shipping cost: $" + shipment1.cost);

System.out.println();

vol = shipment2.volume();

System.out.println("Volume of shipment2 is " + vol);

System.out.println("Weight of shipment2 is "

+ shipment2.weight);

System.out.println("Shipping cost: $" + shipment2.cost);

}

}

2. Write a Java program to illustrate Dynamic method dispatch .

class Figure {

double dim1;

double dim2;

Figure(double a, double b) {

dim1 = a;

dim2 = b;

}

double area() {

System.out.println("Area for Figure is undefined.");

return 0;

}

}

class Rectangle extends Figure {

Rectangle(double a, double b) {

super(a, b);

}

// override area for rectangle

double area() {

System.out.println("Inside Area for Rectangle.");

return dim1 \* dim2;

}

}

class Triangle extends Figure {

Triangle(double a, double b) {

super(a, b);

}

// override area for right triangle

double area() {

System.out.println("Inside Area for Triangle.");

return dim1 \* dim2 / 2;

}

}

public class dynamic {

public static void main(String args[]) {

Figure f = new Figure(10, 10);

Rectangle r = new Rectangle(9, 5);

Triangle t = new Triangle(10, 8);

Figure figref;

figref = r;

System.out.println("Area is " + figref.area());

figref = t;

System.out.println("Area is " + figref.area());

figref = f;

System.out.println("Area is " + figref.area());

}

}

3. Write a Java program to implement fixed stack

interface IntStack {

void push(int item); // store an item

int pop(); // retrieve an item

}

class FixedStack implements IntStack {

private int stck[];

private int tos;

// allocate and initialize stack

FixedStack(int size) {

stck = new int[size];

tos = -1;

}

public void push(int item) {

if(tos==stck.length-1) // use length member

System.out.println("Stack is full.");

else

stck[++tos] = item;

}

// Pop an item from the stack

public int pop() {

if(tos < 0) {

System.out.println("Stack underflow.");

return 0;

}

else

return stck[tos--];

}

}

class IFTest {

public static void main(String args[]) {

FixedStack mystack1 = new FixedStack(5);

FixedStack mystack2 = new FixedStack(8);

for(int i=0; i<5; i++) mystack1.push(i);

for(int i=0; i<8; i++) mystack2.push(i);

System.out.println("Stack in mystack1:");

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

System.out.println(mystack1.pop());

System.out.println("Stack in mystack2:");

for(int i=0; i<8; i++)

System.out.println(mystack2.pop());

}

}

4. Create in interface for a stack. Write a Java program to implement dynamic stack .

interface IntStack {

void push(int item);

int pop();

}

class DynStack implements IntStack {

private int stck[];

private int tos;

DynStack(int size) {

stck = new int[size];

tos = -1;

}

public void push(int item) {

if(tos==stck.length-1) {

int temp[] = new int[stck.length \* 2];

for(int i=0; i<stck.length; i++) temp[i] = stck[i];

stck = temp;

stck[++tos] = item;

}

else

stck[++tos] = item;

}

public int pop() {

if(tos < 0) {

System.out.println("Stack underflow.");

return 0;

}

else

return stck[tos--];

}

}

class IFTest2 {

public static void main(String args[]) {

DynStack mystack1 = new DynStack(5);

DynStack mystack2 = new DynStack(8);

for(int i=0; i<12; i++) mystack1.push(i);

for(int i=0; i<20; i++) mystack2.push(i);

System.out.println("Stack in mystack1:");

for(int i=0; i<12; i++)

System.out.println(mystack1.pop());

System.out.println("Stack in mystack2:");

for(int i=0; i<20; i++)

System.out.println(mystack2.pop());

}

}

5 Write a Java program to illustrate Dynamic method dispatch using abstract classes

abstract class Figure {

double dim1;

double dim2;

Figure(double a, double b) {

dim1 = a;

dim2 = b;

}

abstract double area();

}

class Rectangle extends Figure {

Rectangle(double a, double b) {

super(a, b);

}

double area() {

System.out.println("Inside Area for Rectangle.");

return dim1 \* dim2;

}

}

class Triangle extends Figure {

Triangle(double a, double b) {

super(a, b);

}

double area() {

System.out.println("Inside Area for Triangle.");

return dim1 \* dim2 / 2;

}

}

class AbstractAreas {

public static void main(String args[]) {

Rectangle r = new Rectangle(9, 5);

Triangle t = new Triangle(10, 8);

Figure figref;

figref = r;

System.out.println("Area is " + figref.area());

figref = t;

System.out.println("Area is " + figref.area());

}

}

6. Write a Java program to illustrate scope of different access specifiers using packages ( Create 2 Packages )

Protection.java:

package p1;

public class Protection {

int n = 1;

private int n\_pri = 2;

protected int n\_pro = 3;

public int n\_pub = 4;

public Protection() {

System.out.println("base constructor");

System.out.println("n = " + n);

System.out.println("n\_pri = " + n\_pri);

System.out.println("n\_pro = " + n\_pro);

System.out.println("n\_pub = " + n\_pub);

}

}

This is file Derived.java:

package p1;

class Derived extends Protection {

Derived() {

System.out.println("derived constructor");

System.out.println("n = " + n);

// System.out.println("n\_pri = " + n\_pri);

System.out.println("n\_pro = " + n\_pro);

System.out.println("n\_pub = " + n\_pub);

}

}

This is file SamePackage.java:

package p1;

class SamePackage {

SamePackage() {

Protection p = new Protection();

System.out.println("same package constructor");

System.out.println("n = " + p.n);

// class only

// System.out.println("n\_pri = " + p.n\_pri);

System.out.println("n\_pro = " + p.n\_pro);

System.out.println("n\_pub = " + p.n\_pub);

}

}

This is file Protection2.java:

package p2;

class Protection2 extends p1.Protection {

Protection2() {

System.out.println("derived other package constructor");

// class or package only

// System.out.println("n = " + n);

// class only

// System.out.println("n\_pri = " + n\_pri);

System.out.println("n\_pro = " + n\_pro);

System.out.println("n\_pub = " + n\_pub);

}

}

This is file OtherPackage.java:

package p2;

class OtherPackage {

OtherPackage() {

p1.Protection p = new p1.Protection();

System.out.println("other package constructor");

// class or package only

// System.out.println("n = " + p.n);

// class only

// System.out.println("n\_pri = " + p.n\_pri);

// class, subclass or package only

// System.out.println("n\_pro = " + p.n\_pro);

System.out.println("n\_pub = " + p.n\_pub);

}

}

package p1;

public class Demo {

public static void main(String args[]) {

Protection ob1 = new Protection();

Derived ob2 = new Derived();

SamePackage ob3 = new SamePackage();

}

}

The test file for p2 is shown next:

// Demo package p2.

package p2;

public class Demo {

public static void main(String args[]) {

Protection2 ob1 = new Protection2();

OtherPackage ob2 = new OtherPackage();

}

}

7. Write a Java program to illustrate exceptions (try with multiple catch blocks)

class MultiCatch {

public static void main(String args[]) {

try {

int a = args.length;

System.out.println("a = " + a);

int b = 42 / a;

int c[] = { 1 };

c[42] = 99;

} catch(ArithmeticException e) {

System.out.println("Divide by 0: " + e);

} catch(ArrayIndexOutOfBoundsException e) {

System.out.println("Array index oob: " + e);

}

System.out.println("After try/catch blocks.");

}

}

8. Write a Java program to illustrate exceptions (Nested try/catch blocks)

class NestTry {

public static void main(String args[]) {

try {

int a = args.length;

int b = 42 / a;

System.out.println("a = " + a);

try {

if(a==1)

a = a/(a-a);

if(a==2) {

int c[] = { 1 };

c[42] = 99; // generate an out-of-bounds exception

}

} catch(ArrayIndexOutOfBoundsException e) {

System.out.println("Array index out-of-bounds: " + e);

}

} catch(ArithmeticException e) {

System.out.println("Divide by 0: " + e);

}

}

}

9. Write a Java program to illustrate user defined (custom) exception.

class MyException extends Exception {

private int detail;

MyException(int a) {

detail = a;

}

public String toString() {

return "MyException[" + detail + "]";

}

}

class ExceptionDemo {

static void compute(int a) throws MyException {

System.out.println("Called compute(" + a + ")");

if(a > 10)

throw new MyException(a);

System.out.println("Normal exit");

}

public static void main(String args[]) {

try {

compute(1);

compute(20);

} catch (MyException e) {

System.out.println("Caught " + e);

}

}

}

10 .Develop a Web application (JSP) to display Student Detail form. Make use of a JavaBean to store and retrieve the student details. (usage of action element)

1. JavaBean Class (Student.java)

package beans;

public class Student {

private String name;

private int age;

private String course;

// Getter and Setter for Name

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// Getter and Setter for Age

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

// Getter and Setter for Course

public String getCourse() {

return course;

}

public void setCourse(String course) {

this.course = course;

}

}

2. JSP Form (studentForm.jsp)

<%@ page language="java" contentType="text/html; %>

<html>

<head>

<title>Student Form</title>

</head>

<body>

<h2>Student Detail Form</h2>

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

<label for="name">Name:</label>

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

<label for="age">Age:</label>

<input type="number" id="age" name="age" required><br><br>

<label for="course">Course:</label>

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

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

</form>

</body>

</html>

3. JSP to Display Details (displayStudent.jsp)

<%@ page language="java" contentType="text/html; " %>

<%@ page import="beans.Student" %>

<jsp:useBean id="student" class="beans.Student" scope="request" />

<%

// Setting bean properties using request parameters

student.setName(request.getParameter("name"));

student.setAge(Integer.parseInt(request.getParameter("age")));

student.setCourse(request.getParameter("course"));

%>

<html>

<head>

<title>Student Details</title>

</head>

<body>

<h2>Student Details</h2>

<p><strong>Name:</strong> <%= student.getName() %></p>

<p><strong>Age:</strong> <%= student.getAge() %></p>

<p><strong>Course:</strong> <%= student.getCourse() %></p>

</body>

</html>

11. Web application to illustrate scriptlets (Display Date, odd & even numbers)

<%@ page language="java" contentType="text/html; %>

<html>

<head>

<title>Scriptlet Example</title>

</head>

<body>

<h1>Demonstration of Scriptlets in JSP</h1>

<!-- Display Current Date -->

<h2>Current Date and Time</h2>

<p>

<%

// Scriptlet to display current date and time

java.util.Date date = new java.util.Date();

out.println(date.toString());

%>

</p>

<!-- Display Odd Numbers -->

<h2>First 10 Odd Numbers</h2>

<p>

<%

// Generate and display the first 20 odd numbers

for (int i = 1; i <= 10; i++) {

out.print((2 \* i - 1) + " "); // Odd numbers formula: 2 \* i - 1

}

%>

</p>

<!-- Display Even Numbers -->

<h2>First 10 Even Numbers</h2>

<p>

<%

// Generate and display the first 20 even numbers

for (int i = 1; i <= 10; i++) {

out.print((2 \* i) + " "); // Even numbers formula: 2 \* i

}

%>

</p>

</body>

</html>

11.Web application to illustrate the use of sessions and session tracking

<%@ page language="java" contentType="text/html; %>

<html>

<head>

<title>Scriptlet Example</title>

</head>

<body>

<h1>Demonstration of Scriptlets in JSP</h1>

<!-- Display Current Date -->

<h2>Current Date and Time</h2>

<p>

<%

// Scriptlet to display current date and time

java.util.Date date = new java.util.Date();

out.println(date.toString());

%>

</p>

<!-- Display Odd Numbers -->

<h2>First 10 Odd Numbers</h2>

<p>

<%

// Generate and display the first 20 odd numbers

for (int i = 1; i <= 10; i++) {

out.print((2 \* i - 1) + " "); // Odd numbers formula: 2 \* i - 1

}

%>

</p>

<!-- Display Even Numbers -->

<h2>First 10 Even Numbers</h2>

<p>

<%

// Generate and display the first 20 even numbers

for (int i = 1; i <= 10; i++) {

out.print((2 \* i) + " "); // Even numbers formula: 2 \* i

}

%>

</p>

</body>

</html>

12. Web application to illustrate the use of sessions and session tracking

<%@ page language="java" contentType="text/html; %>

<%@ page import="java.util.Date" %>

<html>

<head>

<title>Session Tracking Example</title>

</head>

<body>

<h1>Welcome to Session Tracking Example</h1>

<%

// Retrieve the current session or create a new one

javax.servlet.http.HttpSession session = request.getSession();

// Check if the user is new or returning

Integer visitCount = (Integer) session.getAttribute("visitCount");

if (visitCount == null) {

// First visit

visitCount = 1;

session.setAttribute("creationTime", new

Date(session.getCreationTime())); // Store session creation time

} else {

// Increment visit count for subsequent visits

visitCount++;

}

// Store updated visit count in session

session.setAttribute("visitCount", visitCount);

// Retrieve session creation time

Date creationTime = (Date) session.getAttribute("creationTime");

// Display session details to the user

%>

<h2>Session Details</h2>

<p><strong>Session ID:</strong> <%= session.getId() %></p>

<p><strong>Session Creation Time:</strong> <%= creationTime %></p>

<p><strong>Number of Visits:</strong> <%= visitCount %></p>

</body>

</html>

13. Web application to illustrate cookies in JSP

Page 1: setCookie.jsp

<%@ page language="java" contentType="text/html; %>

<%@ page import="javax.servlet.http.Cookie" %>

<html>

<head>

<title>Set Cookie Example</title>

</head>

<body>

<h1>Welcome to the Cookie Example - Set Cookie</h1>

<%

// Check if the user is already visiting and has a cookie

Cookie[] cookies = request.getCookies();

boolean hasCookie = false;

if (cookies != null) {

for (Cookie cookie : cookies) {

if ("userName".equals(cookie.getName())) {

hasCookie = true;

break;

}

}

}

if (hasCookie) {

out.println("<p>You already have a cookie. Go to the next page to see the greeting.</p>");

} else {

%>

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

<label for="name">Enter Your Name:</label>

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

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

</form>

<%

}

// Handle form submission to set a cookie

if (request.getMethod().equalsIgnoreCase("POST")) {

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

if (name != null && !name.isEmpty()) {

// Create a new cookie to store the user's name

Cookie userNameCookie = new Cookie("userName", name);

// Set the cookie's maximum age to 1 year (365 days)

userNameCookie.setMaxAge(60 \* 60 \* 24 \* 365);

// Add the cookie to the response

response.addCookie(userNameCookie);

out.println("<p>Cookie has been set! <a href='getCookie.jsp'>Go to the next page to see your greeting.</a></p>");

}

}

%>

</body>

</html>

Page 2: getCookie.jsp

<%@ page language="java" contentType="text/html; %>

<%@ page import="javax.servlet.http.Cookie" %>

<html>

<head>

<title>Get Cookie Example</title>

</head>

<body>

<h1>Welcome to the Cookie Example - Get Cookie</h1>

<%

// Retrieve cookies

Cookie[] cookies = request.getCookies();

String userName = null;

if (cookies != null) {

for (Cookie cookie : cookies) {

if ("userName".equals(cookie.getName())) {

userName = cookie.getValue(); // Get user's name from cookie

break;

}

}

}

if (userName != null) {

out.println("<h2>Hello, " + userName + "!</h2>");

out.println("<p>Welcome back to our site!</p>");

} else {

out.println("<p>No cookie found. Please go back and enter your name first.</p>");

}

%>

</body>

</html>

14. Web application to illustrate the use of core tag library

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

<html>

<head>

<title>JSTL Core Tag Library Example</title>

</head>

<body>

<h1>JSTL Core Tag Library Example</h1>

<!-- 1. Using c:set to set a value -->

<c:set var="userName" value="John Doe" />

<p><strong>User Name:</strong> <c:out value="${userName}" /></p>

<!-- 2. Conditional Statements -->

<c:if test="${userName == 'John Doe'}">

<p>Welcome, John Doe!</p>

</c:if>

<!-- 3. Using c:choose for conditional branching -->

<c:choose>

<c:when test="${userName == 'John Doe'}">

<p>Hello, VIP user!</p>

</c:when>

<c:otherwise>

<p>Hello, Guest!</p>

</c:otherwise>

</c:choose>

<!-- 4. Iterating over a collection using c:forEach -->

<c:set var="items" value="${['Item1', 'Item2', 'Item3', 'Item4']}" />

<h3>List of Items:</h3>

<ul>

<c:forEach var="item" items="${items}">

<li>${item}</li>

</c:forEach>

</ul>

</body>

</html>

15.Web application to illustrate the use of xml tag library

<?xml version="1.0" encoding="UTF-8"?>

<students>

<student>

<id>1</id>

<name>John Doe</name>

<age>21</age>

</student>

<student>

<id>2</id>

<name>Jane Smith</name>

<age>22</age>

</student>

<student>

<id>3</id>

<name>Mike Johnson</name>

<age>23</age>

</student>

</students>

JSP File: xmlExample.jsp

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

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

<html>

<head>

<title>JSTL XML Tag Library Example</title>

</head>

<body>

<!-- Load XML data using c:import -->

<c:import url="data/data.xml" var="xmlData" />

<!-- Parse the XML data -->

<x:parse xml="${xmlData}" var="parsedXml" />

<!-- Display all students using x:forEach -->

<h2>Student List</h2>

<ul>

<x:forEach select="$parsedXml/students/student" var="student">

<li>

<strong>ID:</strong> <x:out select="$student/id" /><br>

<strong>Name:</strong> <x:out select="$student/name" /><br>

<strong>Age:</strong> <x:out select="$student/age" />

</li>

<hr>

</x:forEach>

</ul>

<!-- Display a specific student based on condition -->

<h2>Student with ID 2</h2>

<x:forEach select="$parsedXml/students/student[id=2]" var="student">

<p>

<strong>ID:</strong> <x:out select="$student/id" /><br>

<strong>Name:</strong> <x:out select="$student/name" /><br>

<strong>Age:</strong> <x:out select="$student/age" />

</p>

</x:forEach>

</body>

</html>

**16. Develop a Web application to illustrate action element (Navigation between 3 pages)**

17. Design a Book application form using HTML/JSP. Have input fields for title, author, number of pages and price of the book. There are two buttons SAVE and FIND along with a CLEAR button. When SAVE button is clicked, the form details must be stored to the database (servlet-jdbc). When FIND button is clicked, with only the title of the book entered, retrieve the other details of the book (if available) from the database and display on the screen. Otherwise display an error message and provide a link to the main form

HTML/JSP Form (BookForm.jsp)

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

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<title>Book Application Form</title>

</head>

<body>

<h2>Book Application Form</h2>

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

<label for="title">Title:</label>

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

<label for="author">Author:</label>

<input type="text" id="author" name="author"><br><br>

<label for="pages">Number of Pages:</label>

<input type="number" id="pages" name="pages" min="1"><br><br>

<label for="price">Price:</label>

<input type="number" id="price" name="price" step="0.01"

min="0"><br><br>

<button type="submit" name="action" value="save">SAVE</button>

<button type="submit" name="action" value="find">FIND</button>

<button type="reset">CLEAR</button>

</form>

</body>

</html>

Servlet (BookServlet.java)

import java.io.\*;

import java.sql.\*;

import javax.servlet.\*;

import javax.servlet.http.\*;

import javax.servlet.annotation.WebServlet;

@WebServlet("/BookServlet")

public class BookServlet extends HttpServlet {

private static final String DB\_URL = "jdbc:derby://localhost:1527/bookdb";

private static final String DB\_USER = "root";

private static final String DB\_PASSWORD = "kai";

protected void doPost(HttpServletRequest request, HttpServletResponse

response)

throws ServletException, IOException {

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

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

response.setContentType("text/html");

PrintWriter out = response.getWriter();

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER,

DB\_PASSWORD)) {

if ("save".equals(action)) {

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

int pages = Integer.parseInt(request.getParameter("pages"));

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

String sql = "INSERT INTO books (title, author, pages, price) VALUES (?,

?, ?, ?)";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setString(1, title);

pstmt.setString(2, author);

pstmt.setInt(3, pages);

pstmt.setDouble(4, price);

pstmt.executeUpdate();

out.println("<h3>Book details saved successfully!</h3>");

}

} else if ("find".equals(action)) {

String sql = "SELECT author, pages, price FROM books WHERE title =

?";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setString(1, title);

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

String author = rs.getString("author");

int pages = rs.getInt("pages");

double price = rs.getDouble("price");

out.println("<h3>Book Details:</h3>");

out.println("Title: " + title + "<br>");

out.println("Author: " + author + "<br>");

out.println("Pages: " + pages + "<br>");

out.println("Price: " + price + "<br>");

} else {

out.println("<h3>Book not found!</h3>");

out.println("<a href='BookForm.jsp'>Back to form</a>");

}

}

}

} catch (Exception e) {

out.println("<h3>Error: " + e.getMessage() + "</h3>");

e.printStackTrace(out);

}

}

}

18. Web application to illustrate use of inserting records to a database table (teacher/student) using servlets (jdbc). Also retrieve all the records from the database. Records details have to obtained from a user form.

TeacherForm.jsp

<!DOCTYPE html>

<html>

<head>

<title>Teacher Management</title>

</head>

<body>

<h2>Teacher Form</h2>

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

<label for="name">Name:</label>

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

<label for="subject">Subject:</label>

<input type="text" id="subject" name="subject"><br><br>

<label for="experience">Years of Experience:</label>

<input type="number" id="experience" name="experience"

min="0"><br><br>

<button type="submit" name="action" value="insert">Insert</button>

<button type="submit" name="action" value="view">View All

Records</button>

</form>

</body>

</html>

TeacherServlet.java

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.\*;

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("/TeacherServlet")

public class TeacherServlet extends HttpServlet {

private static final String DB\_URL = "jdbc:derby://localhost:1527/bookdb";

private static final String DB\_USER = "root";

private static final String DB\_PASSWORD = "kai";

protected void doPost(HttpServletRequest request, HttpServletResponse

response)

throws ServletException, IOException {

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

response.setContentType("text/html");

PrintWriter out = response.getWriter();

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER,

DB\_PASSWORD)) {

if ("insert".equals(action)) {

// Retrieve form data

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

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

int years = Integer.parseInt(request.getParameter("experience"));

// Insert into the Teacher table

String sql = "INSERT INTO TEACHER (name, subject, years) VALUES (?,

?, ?)";

try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

pstmt.setString(1, name);

pstmt.setString(2, subject);

pstmt.setInt(3, years);

pstmt.executeUpdate();

out.println("<h3>Teacher record inserted successfully!</h3>");

}

} else if ("view".equals(action)) {

// Retrieve all records from the Teacher table

String sql = "SELECT \* FROM TEACHER";

try (Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

out.println("<h3>All Teacher Records:</h3>");

out.println("<table border='1'>");

out.println("<tr><th>ID</th><th>Name</th><th>Subject</th><th>Experience

</th></tr>");

while (rs.next()) {

out.println("<tr>");

out.println("<td>" + rs.getInt("id") + "</td>");

out.println("<td>" + rs.getString("name") + "</td>");

out.println("<td>" + rs.getString("subject") + "</td>");

out.println("<td>" + rs.getInt("years") + "</td>");

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

}

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

}

}

} catch (Exception e) {

out.println("<h3>Error: " + e.getMessage() + "</h3>");

e.printStackTrace(out);

}

}

}

19. Develop an Enterprise application to illustrate the use of stateful session bean (CartBean) (with local interface and Jsp)

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

<%@page import="bean.\*;" %>

<!DOCTYPE html>

<html>

<body>

<%!

private CartBeanLocal ct;

public void jspInit() {

ct = new CartBean();

}

public void jspDestroy() {

ct = null;

}

%>

<h1>Shopping Cart</h1>

<form>

Enter An Item: <input type="text" name="item" value="itme" required >

<input type="submit" name="add" value="add" >

</form>

<%

String st = request.getParameter("item");

if(st != null)

ct.addItem(st);

out.print(ct.getItem());

%>

</body>

</html>

CartBean.java

package bean;

import java.util.ArrayList;

import java.util.Collection;

import javax.ejb.Stateful;

@Stateful

public class CartBean implements CartBeanLocal {

private ArrayList itm;

public CartBean() {

itm = new ArrayList();

}

@Override

public void addItem(String item) {

itm.add(item);

}

@Override

public Collection getItem() {

return itm;

}

}

CartBeanLocal.java

package bean;

import java.util.Collection;

import javax.ejb.Local;

@Local

public interface CartBeanLocal {

void addItem(String item);

Collection getItem();

}

20 Develop an Enterprise application to illustrate the use of stateless (calculator) session bean, local interface

Calculator:

Calc.java

package beans;

import javax.ejb.Stateless;

@Stateless

public class calc implements calcLocal {

@Override

public double add(double num1,double num2) {

return num1+num2;

}

@Override

public double sub(double num1, double num2) {

return num1-num2;

}

public double mul(double num1, double num2) {

return num1\*num2;

}

public double div(double num1, double num2) {

return num1/num2;

}

}

CalcLocal.java

package beans;

import javax.ejb.Local;

@Local

public interface calcLocal {

double add(double num1,double num2);

double sub(double num1, double num2);

double mul(double num1, double num2);

double div(double num1, double num2);

}

Calc1.jsp

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

<%@page import="beans.\*;" %>

<!DOCTYPE html>

<html>

<head>

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

<title>JSP Page</title>

</head>

<%!

private calcLocal c1;

public void jspInit(){

c1=new calc();

}

public void jspDistroy(){

c1=null;

}

%>

<body>

<h1>Result</h1>

<%

double n1=Double.parseDouble(request.getParameter("n1"));

double n2=Double.parseDouble(request.getParameter("n2"));

int labl=Integer.parseInt(request.getParameter("op"));

switch(labl){

case 1:

out.println(c1.add(n1,n2));

break;

case 2:

out.println(c1.sub(n1,n2));

break;

case 3:

out.println(c1.mul(n1,n2));

break;

case 4:

if(n2==0)

{

out.println("Zero division Not possible.");

}else{

out.println(c1.div(n1,n2));

}

break;

}

%>

</body>

</html>

Calc2.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>Calculator</h1>

<form action="cal1.jsp">

<input type="text" name="n1" value="" placeholder="enter num1" />

<input type="text" name="n2" value="" placeholder="enter num2"/>

<button value="1" name="op" >+</button>

<button value="2" name="op" >-</button>

<button value="3" name="op" >\*</button>

<button value="4" name="op" >/</button>

</form>

</body>

</html>