UNIVERSITY OF MUMBAI

CENTER FOR DISTANCE AND OPEN EDUCATION (CDOE)

CERTIFICATE

THE EXPERIMENTS DULY SIGNED IN THIS JOURNAL REPRESENT THE WORK DONE BY MST./MS Shilpa Bhosale APPLICATION ID.: 20672 & SEAT NO.: 8221054 RESPECTIVELY IN **SEMESTER I** OF **FIRST YEAR OF MASTER OF COMPUTER APPLICATION (NEP)** IN THE COMPUTER LABORATORY OF PCP CENTER Prahladrai Dalmia Lions College FOR SUBJECT Advance Java DURING THE ACADEMIC YEAR 2025-2026.

SUBJECT INCHARGE	HEAD OF DEPARTMENT	EXTERNAL EXAMINER

Sr no.	Topic	Date	Remark
1	1. Assignments on Java Generics a) Write a Java Program to demonstrate a Generic Class. b) Write a Java Program to demonstrate Generic Methods. c) Write a Java Program to demonstrate Wildcards in Java Generics.		
2	2. Assignments on List Interface a) Write a Java program to create List containing list of items of type b) String and use for-each loop to print the items of the list. c) b. Write a Java program to create List containing list of items & use ListIterator interface to print items present in the list. Also print the list in reverse / backward direction.		
3	3. Assignments on Set Interface a) Write a Java program to create a Set containing list of items of type String and print the items in the list using Iterator interface. Also print the list in reverse / backward direction. b) Write a Java program using Set interface containing list of items and perform the following operations: a. Add items in the set. b. Insert items of one set in to other set. c. Remove items from the set d. d. Search the specified item in the set		
4	4. Assignments on Map Interface a) Write a Java program using Map interface containing list of items having keys and associated values and perform the following operations: a. Add items in the map. b. Remove items from the map c. Search specific key from the map d. Get value of the specified key e. Insert map elements of one map in to other map. f. f. Print all keys and values of the map.		

5	5 Assignments on Lambda Expression a) WAP using Lambda Expression to print "Hello World". b) WAP using Lambda Expression with single parameters. c) Write a Java program using Lambda Expression with multiple parameters to add two numbers. d) Write a Java program using Lambda Expression to calculate the following: e) Write a Java program using Lambda Expression with or without return keyword. f) Write a Java program using Lambda Expression to concatenate two strings.		
6	6. Assignments based on web application development using JSP a) Create a Telephone directory using JSP and store all the information within a database, so that later could be retrieved as per the requirement. Make your own assumptions. b) Write a JSP page to display the Registration form (Make your own assumptions) c) Write a JSP program to add, delete and display the records from StudentMaster (RollNo, Name, Semester, Course) table. d) Design Loan calculator using JSP which accepts Period of Time (in years) and Principal Loan Amount. Display the payment amount for each loan and then list the loan balance and interest paid for each payment over the term of the loan for the following time period and interest rate: a. 1 to 7 year at 5.35% b. 8 to 15 year at 5.5% c. 16 to 30 year at 5.75% e) Write a program using JSP that displays a webpage consisting an Application form for change of Study Center which can be filled by any student who wants to change his/ her study center. Make necessary assumptions. f) Write a JSP program that demonstrates the use of JSP declaration, scriptlet, directives, expression, header and footer.		
7	7. Assignment based Spring Framework a) Write a program to print "Hello World" using spring framework. b) Write a program to demonstrate dependency injection via setter method. c) Write a program to demonstrate dependency injection via Constructor. d) Write a program to demonstrate Autowiring.		

8	8. Assignment based Aspect Oriented Programming a) Write a program to demonstrate Spring AOP – before advice. b) Write a program to demonstrate Spring AOP – after advice. c) Write a program to demonstrate Spring AOP – around advice. d) Write a program to demonstrate Spring AOP – after returning advice. Write a program to demonstrate Spring AOP – after throwing advice. Write a program to demonstrate Spring AOP – pointcuts.		

1. Assignments on Java Generics

a. Write a Java Program to demonstrate a Generic Class.

Code:

```
class GenericBox<T> {
    private T value;

    public GenericBox(T value) {
        this.value = value;
    }

    public void setValue(T value) {
        this.value = value;
    }

    public T getValue() {
        return value;
    }

    public void display() {
        System.out.println("Stored value: " + value);
    }
}

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

        GenericBox<Integer> intBox = new GenericBox<>(100);
        intBox.display();

        GenericBox<String> strBox = new GenericBox<>("Hello Generics");
```

```
        strBox.display();

        GenericBox<Double> doubleBox = new GenericBox<>(45.67);
        doubleBox.display();
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac GenericClassDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java GenericClassDemo
Stored value: 100
Stored value: Hello Generics
Stored value: 45.67
```

b. Write a Java Program to demonstrate Generic Methods.

Code:

```
class GenericMethodDemo {

    public static <T> void display(T value) {
        System.out.println("Value: " + value);
    }

    public static <T> void printArray(T[] array) {
        for (T element : array) {
            System.out.println(element);
        }
    }

    public static void main(String[] args) {

        display(10);
        display("Hello Generic Method");
        display(25.5);

        Integer[] intArray = {1, 2, 3};
        String[] strArray = {"Java", "Generics", "Method"};

        printArray(intArray);
        printArray(strArray);
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)\Practical 2>javac GenericMethodDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)\Practical 2>java GenericMethodDemo
Value: 10
Value: Hello Generic Method
Value: 25.5
1
2
3
Java
Generics
Method
```

c. Write a Java Program to demonstrate Wildcards in Java Generics.

Code:

```
import java.util.*;

class WildcardDemo {

    public static void printList(List<?> list) {
        for (Object element : list) {
            System.out.println(element);
        }
    }

    public static void printNumbers(List<? extends Number> list) {
        for (Number num : list) {
            System.out.println(num);
        }
    }

    public static void addNumbers(List<? super Integer> list) {
        list.add(10);
        list.add(20);
    }

    public static void main(String[] args) {

        List<Integer> intList = Arrays.asList(1, 2, 3);
        List<Double> doubleList = Arrays.asList(1.5, 2.5, 3.5);
        List<Number> numberList = new ArrayList<>();

        System.out.println("Unbounded Wildcard:");
```

```
        printList(intList);
        printList(doubleList);

        System.out.println("\nUpper Bounded Wildcard:");
        printNumbers(intList);
        printNumbers(doubleList);

        System.out.println("\nLower Bounded Wildcard:");
        addNumbers(numberList);
        printList(numberList);
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac WildcardDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java WildcardDemo
Unbounded Wildcard:
1
2
3
1.5
2.5
3.5

Upper Bounded Wildcard:
1
2
3
1.5
2.5
3.5

Lower Bounded Wildcard:
10
20
```

2. Assignments on List Interface

a. Write a Java program to create List containing list of items of type

Code:

```
import java.util.*;


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


      List<String> items = new ArrayList<>();


      items.add("Pen");
      items.add("Notebook");
      items.add("Pencil");
      items.add("Eraser");


      System.out.println("List of Items:");
      for (String item : items) {
         System.out.println(item);
      }
   }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac ListDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java ListDemo
List of Items:
Pen
Notebook
Pencil
Eraser
```

b. Write a Java program to create List containing list of items & use ListIterator interface to print items present in the list. Also print the list in reverse / backward direction.

Code:

```
import java.util.*;


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

        List<String> items = new ArrayList<>();


        items.add("Laptop");
        items.add("Mobile");
        items.add("Tablet");
        items.add("Headphones");


        System.out.println("Forward Direction:");
        ListIterator<String> itr = items.listIterator();
        while (itr.hasNext()) {
            System.out.println(itr.next());
        }


        System.out.println("\nReverse Direction:");
        while (itr.hasPrevious()) {
            System.out.println(itr.previous());
        }
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac ListIteratorDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java ListIteratorDemo
Forward Direction:
Laptop
Mobile
Tablet
Headphones

Reverse Direction:
Headphones
Tablet
Mobile
Laptop
```

3. Assignments on Set Interface

a. Write a Java program to create a Set containing list of items of type String and print the items in the list using Iterator interface. Also print the list in reverse / backward direction.

Code:

```
import java.util.*;


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


      Set<String> items = new LinkedHashSet<>();


      items.add("Pen");
      items.add("Book");
      items.add("Bag");
      items.add("Bottle");


      System.out.println("Forward Direction:");
      Iterator<String> itr = items.iterator();
      while (itr.hasNext()) {
         System.out.println(itr.next());
      }


      System.out.println("\nReverse Direction:");
      List<String> list = new ArrayList<>(items);
      ListIterator<String> listItr = list.listIterator(list.size());
      while (listItr.hasPrevious()) {
         System.out.println(listItr.previous());
      }
```

```
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac SetIteratorDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java SetIteratorDemo
Forward Direction:
Pen
Book
Bag
Bottle

Reverse Direction:
Bottle
Bag
Book
Pen
```

b. Write a Java program using Set interface containing list of items and perform the following operations:

a. Add items in the set.

b. Insert items of one set in to other set.

c. Remove items from the set

d. Search the specified item in the set

Code:

```
import java.util.*;


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


      Set<String> set1 = new HashSet<>();


      set1.add("Apple");
      set1.add("Banana");
      set1.add("Mango");
      set1.add("Orange");


      System.out.println("Set 1: " + set1);


      Set<String> set2 = new HashSet<>();
      set2.add("Grapes");
      set2.add("Pineapple");


      set1.addAll(set2);
      System.out.println("After inserting Set2 into Set1: " + set1);


      set1.remove("Banana");
```

```
        System.out.println("After removing Banana: " + set1);

        String searchItem = "Mango";
        if (set1.contains(searchItem)) {
            System.out.println(searchItem + " is found in the set.");
        } else {
            System.out.println(searchItem + " is not found in the set.");
        }
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac SetOperationsDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java SetOperationsDemo
Set 1: [Apple, Mango, Orange, Banana]
After inserting Set2 into Set1: [Apple, Grapes, Mango, Pineapple, Orange, Banana]
After removing Banana: [Apple, Grapes, Mango, Pineapple, Orange]
Mango is found in the set.
```

4. Assignments on Map Interface

a. Write a Java program using Map interface containing list of items having keys and associated values and perform the following operations:

a. Add items in the map.

b. Remove items from the map

c. Search specific key from the map

d. Get value of the specified key

e. Insert map elements of one map in to other map.

f. Print all keys and values of the map.

Code:

```
import java.util.*;


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


      Map<Integer, String> map1 = new HashMap<>();


      map1.put(1, "Apple");
      map1.put(2, "Banana");
      map1.put(3, "Mango");
      map1.put(4, "Orange");


      System.out.println("Map 1: " + map1);


      map1.remove(2);
      System.out.println("After removing key 2: " + map1);


      int searchKey = 3;
```

```
        if (map1.containsKey(searchKey)) {
            System.out.println("Key " + searchKey + " is found in the map.");
        } else {
            System.out.println("Key " + searchKey + " is not found in the map.");
        }

        System.out.println("Value for key " + searchKey + ": " + map1.get(searchKey));

        Map<Integer, String> map2 = new HashMap<>();
        map2.put(5, "Grapes");
        map2.put(6, "Pineapple");

        map1.putAll(map2);
        System.out.println("After inserting Map2 into Map1: " + map1);

        System.out.println("All keys and values:");
        for (Map.Entry<Integer, String> entry : map1.entrySet()) {
            System.out.println("Key: " + entry.getKey() + ", Value: " + entry.getValue());
        }
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac MapOperationsDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java MapOperationsDemo
Map 1: {1=Apple, 2=Banana, 3=Mango, 4=Orange}
After removing key 2: {1=Apple, 3=Mango, 4=Orange}
Key 3 is found in the map.
Value for key 3: Mango
After inserting Map2 into Map1: {1=Apple, 3=Mango, 4=Orange, 5=Grapes, 6=Pineapple}
All keys and values:
Key: 1, Value: Apple
Key: 3, Value: Mango
Key: 4, Value: Orange
Key: 5, Value: Grapes
Key: 6, Value: Pineapple
```

5. Assignments on Lambda Expression

a. WAP using Lambda Expression to print “Hello World”.

Code:

```
interface Message {
    void print();
}

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

        Message msg = () -> System.out.println("Hello World");
        msg.print();
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac LambdaHelloWorld.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java LambdaHelloWorld
Hello World
```

b. WAP using Lambda Expression with single parameters.

Code:

```
 interface Display {
    void show(String message);
}


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


        Display d = message -> System.out.println(message);
        d.show("Welcome to Java Lambda Expression");
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac LambdaSingleParameter.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java LambdaSingleParameter
Welcome to Java Lambda Expression
```

c. Write a Java program using Lambda Expression with multiple parameters to add two numbers.

Code:

```
interface Addition {
    int add(int a, int b);
}

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

        Addition sum = (a, b) -> a + b;
        System.out.println("Sum = " + sum.add(10, 20));
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac LambdaMultipleParameters.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java LambdaMultipleParameters
Sum = 30
```

d. Write a Java program using Lambda Expression to calculate the following:

a. Convert Fahrenheit to Celsius

b. Convert Kilometers to Miles.

Code:

```
interface Converter {
   double convert(double value);
}


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


      Converter fahrenheitToCelsius = f ->
            (f - 32) * 5 / 9;


      Converter kilometersToMiles = km ->
            km * 0.621371;


      System.out.println("Fahrenheit to Celsius:");
      System.out.println("98.6 F = " + fahrenheitToCelsius.convert(98.6) + " C");


      System.out.println("\nKilometers to Miles:");
      System.out.println("10 Km = " + kilometersToMiles.convert(10) + " Miles");
   }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac LambdaConversionDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java LambdaConversionDemo
Fahrenheit to Celsius:
98.6 F = 37.0 C

Kilometers to Miles:
10 Km = 6.21371 Miles
```

e. Write a Java program using Lambda Expression with or without return keyword.

Code:

```
interface Operation {
    int calculate(int a, int b);
}


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

        Operation addWithReturn = (a, b) -> {
            return a + b;
        };

        Operation addWithoutReturn = (a, b) -> a + b;

        System.out.println("With return keyword: " + addWithReturn.calculate(10, 20));
        System.out.println("Without return keyword: " + addWithoutReturn.calculate(30, 40));
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac LambdaReturnDemo.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java LambdaReturnDemo
With return keyword: 30
Without return keyword: 70
```

f. Write a Java program using Lambda Expression to concatenate two strings.

Code:

```
interface StringConcat {
    String combine(String s1, String s2);
}


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


        StringConcat concat = (s1, s2) -> s1 + s2;
        System.out.println("Result: " + concat.combine("Hello ", "World"));
    }
}
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>javac LambdaStringConcat.java

C:\Users\ASUS\OneDrive\Desktop\Java ASIGN(SHIlPA)>java LambdaStringConcat
Result: Hello World
```

6. Assignments based on web application development using JSP

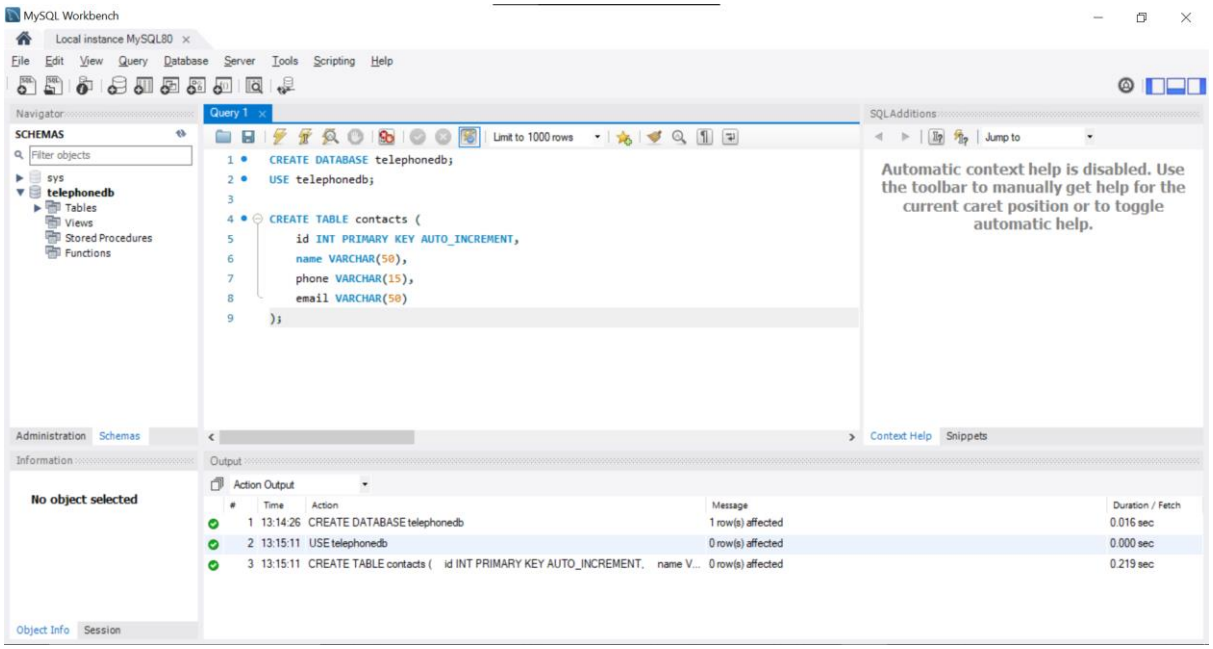
a. Create a Telephone directory using JSP and store all the information within a database, so that later could be retrieved as per the requirement. Make your own assumptions.

Code:

Database Creation (MySQL):

```
CREATE DATABASE telephonedb;


USE telephonedb;


CREATE TABLE contacts (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50),
    phone VARCHAR(15),
    email VARCHAR(50)
);
```

Output:

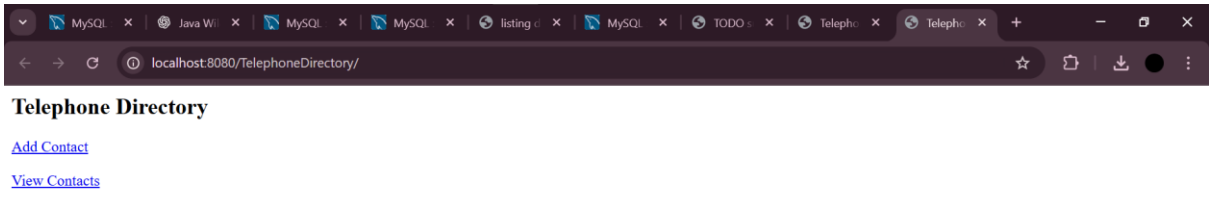


Code:

Index.jsp

```
<html>
<head>
    <title>Telephone Directory</title>
</head>
<body>
    <h2>Telephone Directory</h2>
    <a href="addContact.jsp">Add Contact</a><br><br>
    <a href="viewContacts.jsp">View Contacts</a>
</body>
</html>
```

Output:

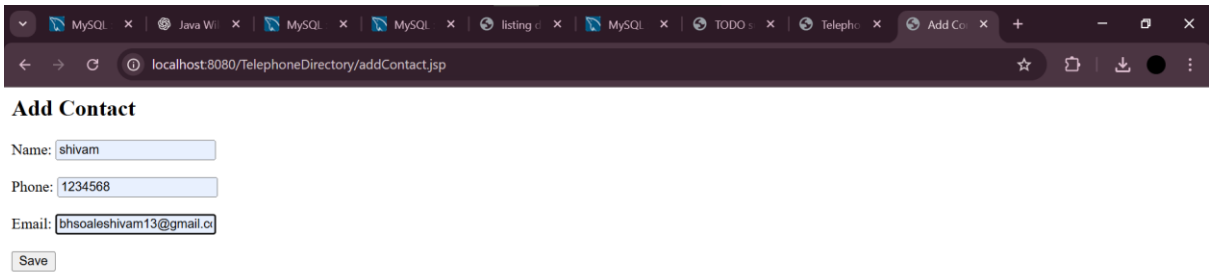


Code:

saveContact.java

```
<%@ page language="java" %>
<html>
<head>
    <title>Add Contact</title>
</head>
<body>
    <h2>Add Contact</h2>
    <form action="saveContact.jsp" method="post">
        Name: <input type="text" name="name" required><br><br>
        Phone: <input type="text" name="phone" required><br><br>
        Email: <input type="email" name="email"><br><br>
        <input type="submit" value="Save">
    </form>
</body>
</html>
```

Output:



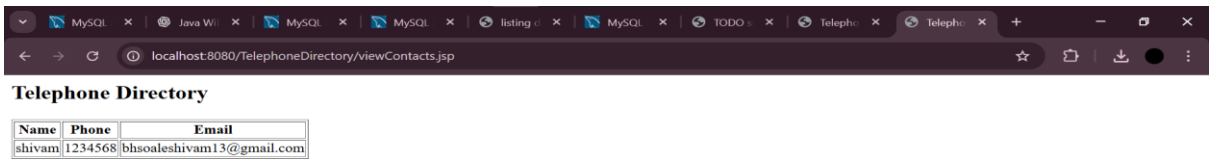
Code:

ViewContact.jsp

```
<%@ page import="java.sql.*" %>

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

Connection con = DriverManager.getConnection(
    "jdbc:mysql://localhost:3306/telephonedb?useSSL=false&allowPublicKeyRetrieval=true",
    "root",
    "1234"
);

Statement st = con.createStatement();
ResultSet rs = st.executeQuery("SELECT * FROM contacts");
%>

<html>
<head>
    <title>Telephone Directory</title>
</head>
<body>

<h2>Telephone Directory</h2>

<table border="1">
<tr>
    <th>Name</th>
    <th>Phone</th>
    <th>Email</th>
```

```
</tr>

<%
while(rs.next()) {
%>
<tr>
    <td><%= rs.getString("name") %></td>
    <td><%= rs.getString("phone") %></td>
    <td><%= rs.getString("email") %></td>
</tr>
<%
}

con.close();
%>

</table>

</body>
</html>
```

Output:

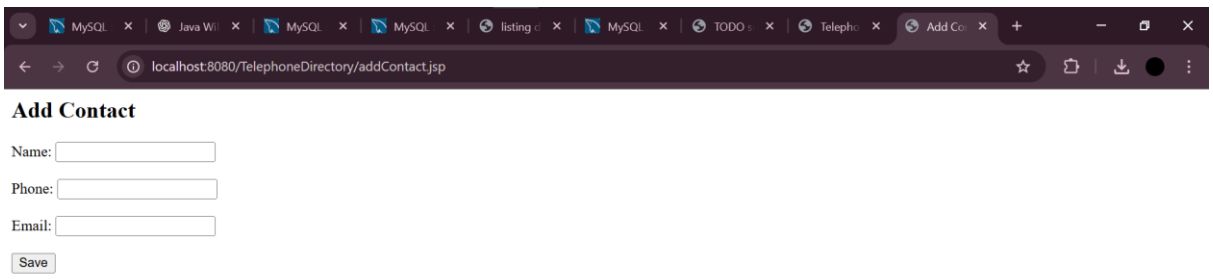


Code:

addContact.jsp

```
<%@ page language="java" %>
<html>
<head>
    <title>Add Contact</title>
</head>
<body>
    <h2>Add Contact</h2>
    <form action="saveContact.jsp" method="post">
        Name: <input type="text" name="name" required><br><br>
        Phone: <input type="text" name="phone" required><br><br>
        Email: <input type="email" name="email"><br><br>
        <input type="submit" value="Save">
    </form>
</body>
</html>
```

Output:



b. Write a JSP page to display the Registration form (Make your own assumptions)

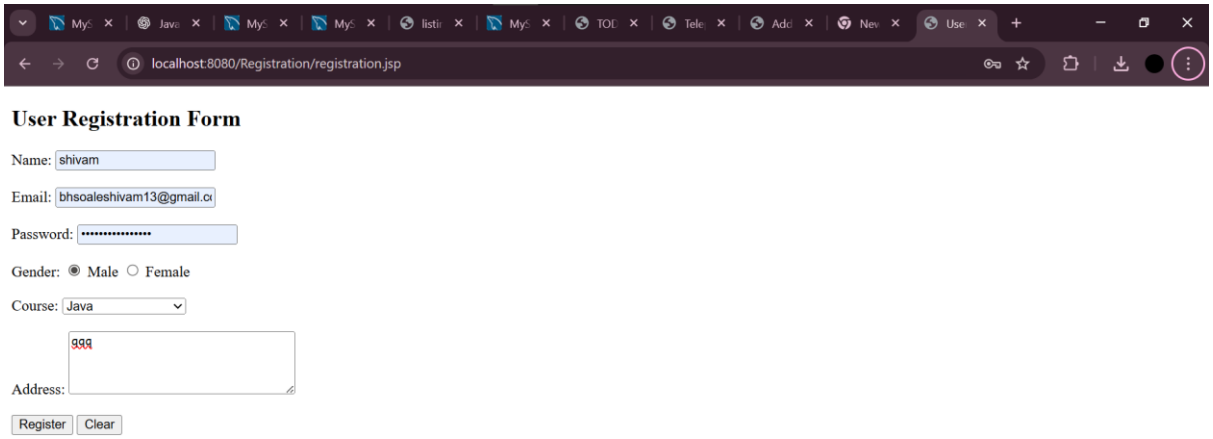
Code:

register.jsp

```
<%@ page language="java" %>
<!DOCTYPE html>
<html>
<head>
    <title>Registration Details</title>
</head>
<body>

<h2>Registration Successful</h2>

<%
    String name = request.getParameter("name");
    String email = request.getParameter("email");
    String password = request.getParameter("password");
    String gender = request.getParameter("gender");
    String course = request.getParameter("course");
    String address = request.getParameter("address");
%>

<table border="1" cellpadding="5">
    <tr>
        <th>Name</th>
        <td><%= name %></td>
    </tr>
```

```
    <tr>
        <th>Email</th>
        <td><%= email %></td>
    </tr>
    <tr>
        <th>Password</th>
        <td><%= password %></td>
    </tr>
    <tr>
        <th>Gender</th>
        <td><%= gender %></td>
    </tr>
    <tr>
        <th>Course</th>
        <td><%= course %></td>
    </tr>
    <tr>
        <th>Address</th>
        <td><%= address %></td>
    </tr>
</table>

<br>
<a href="registration.jsp">Go Back</a>

</body>
</html>
```

Output:



Code:

Registeration.jsp

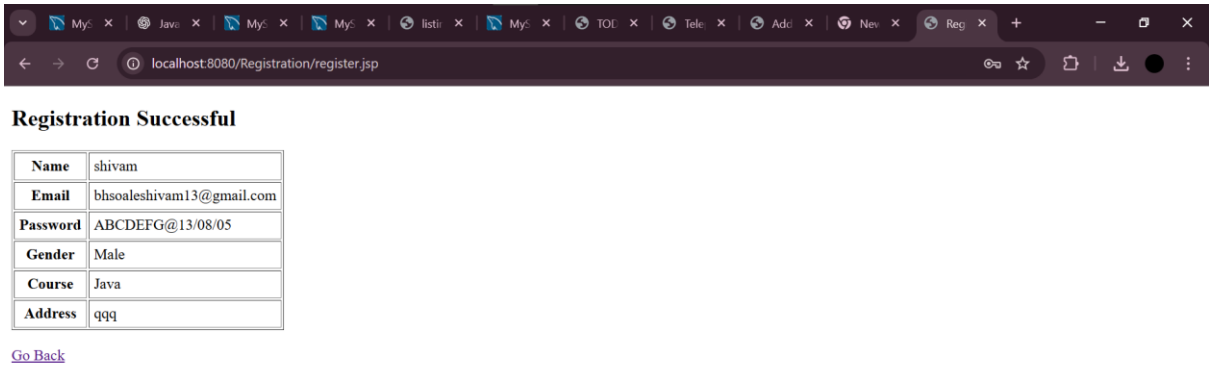
```
<%@ page language="java" %>
<!DOCTYPE html>
<html>
<head>
    <title>User Registration</title>
</head>
<body>

<h2>User Registration Form</h2>

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

    Name:
    <input type="text" name="name" required>
    <br><br>
```

```
    Email:
    <input type="email" name="email" required>
    <br><br>

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

    Gender:
    <input type="radio" name="gender" value="Male"> Male
    <input type="radio" name="gender" value="Female"> Female
    <br><br>

    Course:
    <select name="course">
        <option value="Java">Java</option>
        <option value="Python">Python</option>
        <option value="Web Development">Web Development</option>
    </select>
    <br><br>

    Address:
    <textarea name="address" rows="4" cols="30"></textarea>
    <br><br>

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

</form>
```

</body>

</html>

Output:

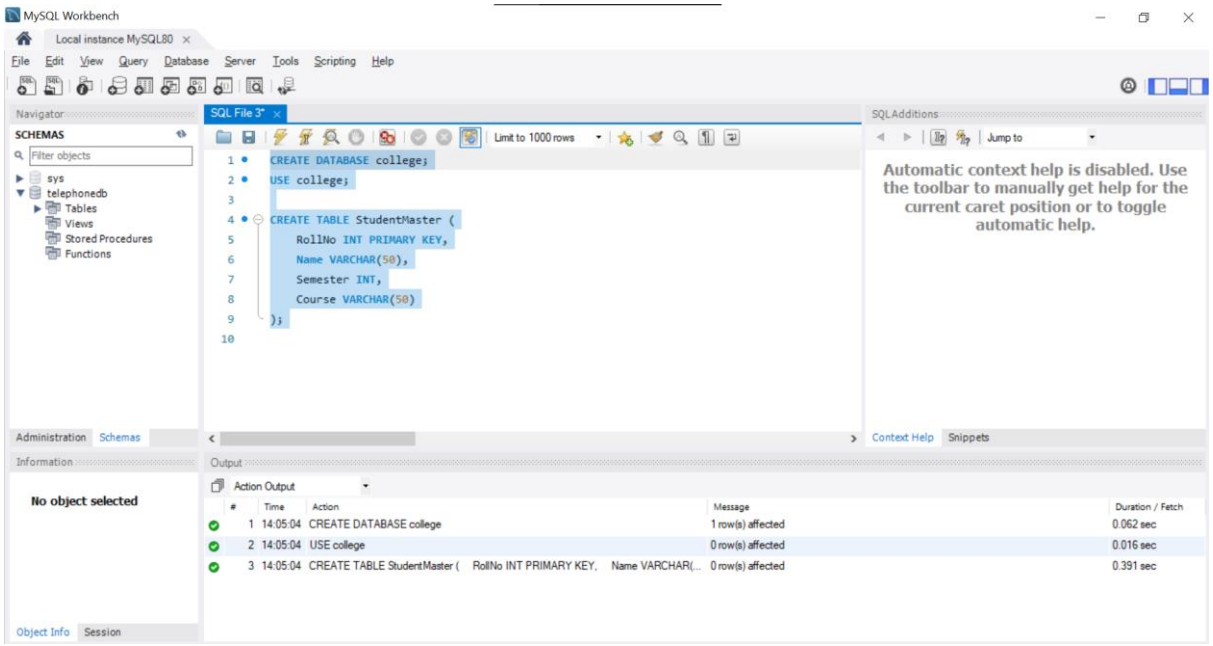


c. Write a JSP program to add, delete and display the records from StudentMaster (RollNo, Name, Semester, Course) table.

Code (MySQL):

```
CREATE DATABASE college;

USE college;


CREATE TABLE StudentMaster (
    RollNo INT PRIMARY KEY,
    Name VARCHAR(50),
    Semester INT,
    Course VARCHAR(50)
);
```

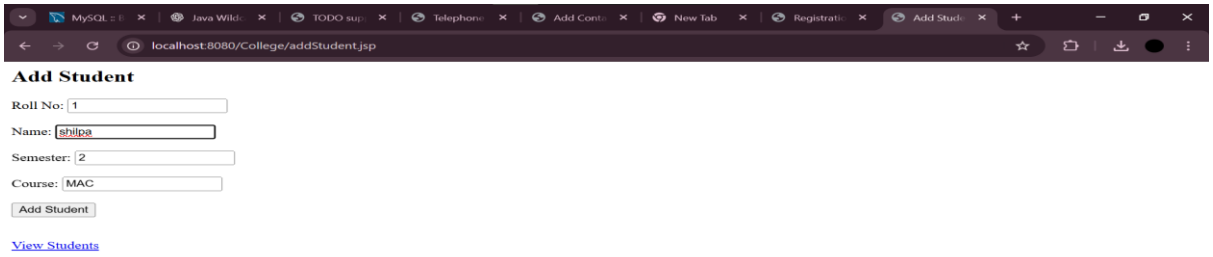
Output:



Code (addStudent.jsp):

```
<html>
<head>
    <title>Add Student</title>
</head>
<body>

<h2>Add Student</h2>

<form action="insertStudent.jsp" method="post">
    Roll No: <input type="number" name="rollno" required><br><br>
    Name: <input type="text" name="name" required><br><br>
    Semester: <input type="number" name="semester" required><br><br>
    Course: <input type="text" name="course" required><br><br>
    <input type="submit" value="Add Student">
</form>

<br>
<a href="viewStudents.jsp">View Students</a>

</body>
</html>
```

Output:


localhost:8080/College/addStudent.jsp
Add Student
Roll No: 1
Name: shilpa
Semester: 2
Course: MAC
Add Student
View Students

Code (insertStudent.jsp):

```
<%@ page import="java.sql.*" %>

<%
int rollno = Integer.parseInt(request.getParameter("rollno"));
String name = request.getParameter("name");
int semester = Integer.parseInt(request.getParameter("semester"));
String course = request.getParameter("course");

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

Connection con = DriverManager.getConnection(
 "jdbc:mysql://localhost:3306/college?useSSL=false&allowPublicKeyRetrieval=true",
 "root",
 "your_password"
);

PreparedStatement ps = con.prepareStatement(
 "INSERT INTO StudentMaster VALUES (?,?,?,?)");

ps.setInt(1, rollno);
ps.setString(2, name);
ps.setInt(3, semester);
ps.setString(4, course);

ps.executeUpdate();
con.close();
%>

<h3>Student Added Successfully</h3>
```

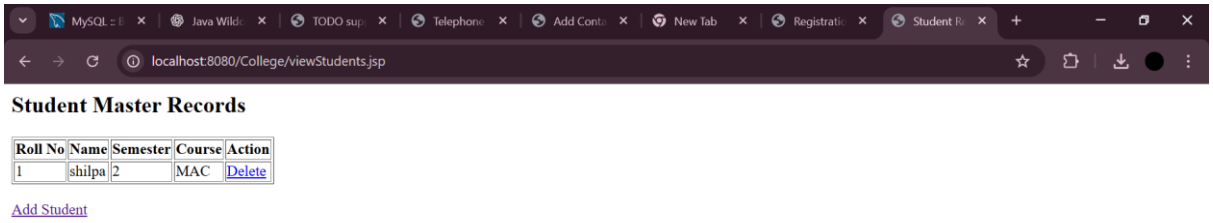
```
<a href="addStudent.jsp">Add More</a> |
<a href="viewStudents.jsp">View Students</a>
```

Code (viewStudents.jsp):

```
<%@ page import="java.sql.*" %>

<html>
<head>
   <title>Student Records</title>
</head>
<body>

<h2>Student Master Records</h2>

<table border="1">
<tr>
   <th>Roll No</th>
   <th>Name</th>
   <th>Semester</th>
   <th>Course</th>
   <th>Action</th>
</tr>

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

Connection con = DriverManager.getConnection(
 "jdbc:mysql://localhost:3306/college?useSSL=false&allowPublicKeyRetrieval=true",
 "root",
 "your_password"
);

Statement st = con.createStatement();
```

```
ResultSet rs = st.executeQuery("SELECT * FROM StudentMaster");

while(rs.next()) {
%>
<tr>
    <td><%= rs.getInt("RollNo") %></td>
    <td><%= rs.getString("Name") %></td>
    <td><%= rs.getInt("Semester") %></td>
    <td><%= rs.getString("Course") %></td>
    <td>
        <a href="deleteStudent.jsp?rollno=<%= rs.getInt("RollNo") %>">
            Delete
        </a>
    </td>
</tr>
<%
}
con.close();
%>

</table>

<br>
<a href="addStudent.jsp">Add Student</a>

</body>
</html>
```

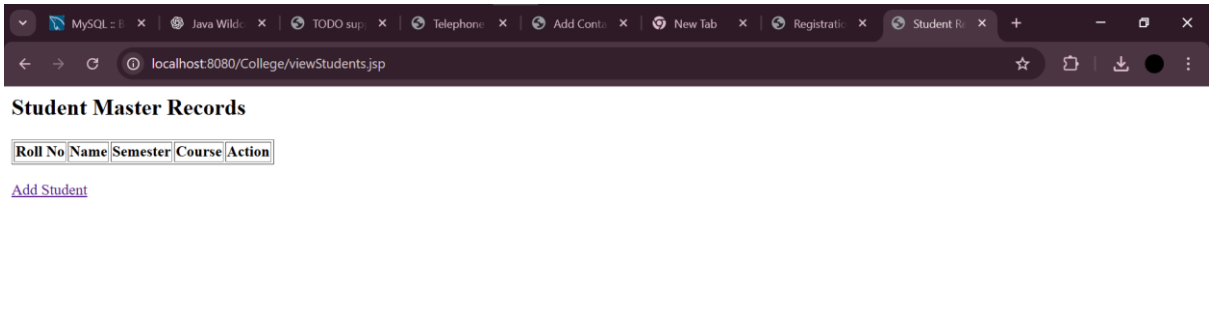
Output:


localhost:8080/College/viewStudents.jsp
Student Master Records
Roll No
Name
Semester
Course
Action
1
shilpa
2
MAC
Delete
Add Student

Code (deleteStudent.jsp):

```
<%@ page import="java.sql.*" %>

<%
int rollno = Integer.parseInt(request.getParameter("rollno"));

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

Connection con = DriverManager.getConnection(
 "jdbc:mysql://localhost:3306/college?useSSL=false&allowPublicKeyRetrieval=true",
 "root",
 "your_password"
);

PreparedStatement ps = con.prepareStatement(
 "DELETE FROM StudentMaster WHERE RollNo=?");

ps.setInt(1, rollno);
ps.executeUpdate();
con.close();

response.sendRedirect("viewStudents.jsp");
%>
```

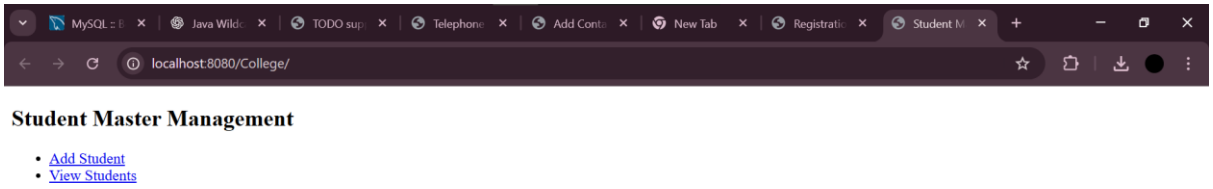
Output:


localhost:8080/College/viewStudents.jsp
Student Master Records
Roll No
Name
Semester
Course
Action
Add Student

Code(Index.jsp):

```
<!DOCTYPE html>
<html>
<head>
    <title>Student Master System</title>
</head>
<body>

<h2>Student Master Management</h2>

<ul>
    <li><a href="addStudent.jsp">Add Student</a></li>
    <li><a href="viewStudents.jsp">View Students</a></li>
</ul>

</body>
</html>
```

Output:



d. Design Loan calculator using JSP which accepts Period of Time (in years) and Principal Loan Amount. Display the payment amount for each loan and then list the loan balance and interest paid for each payment over the term of the loan for the following time period and interest rate:

a) 1 to 7 year at 5.35%

b) 8 to 15 year at 5.5%

c) 16 to 30 year at 5.75%

Code:

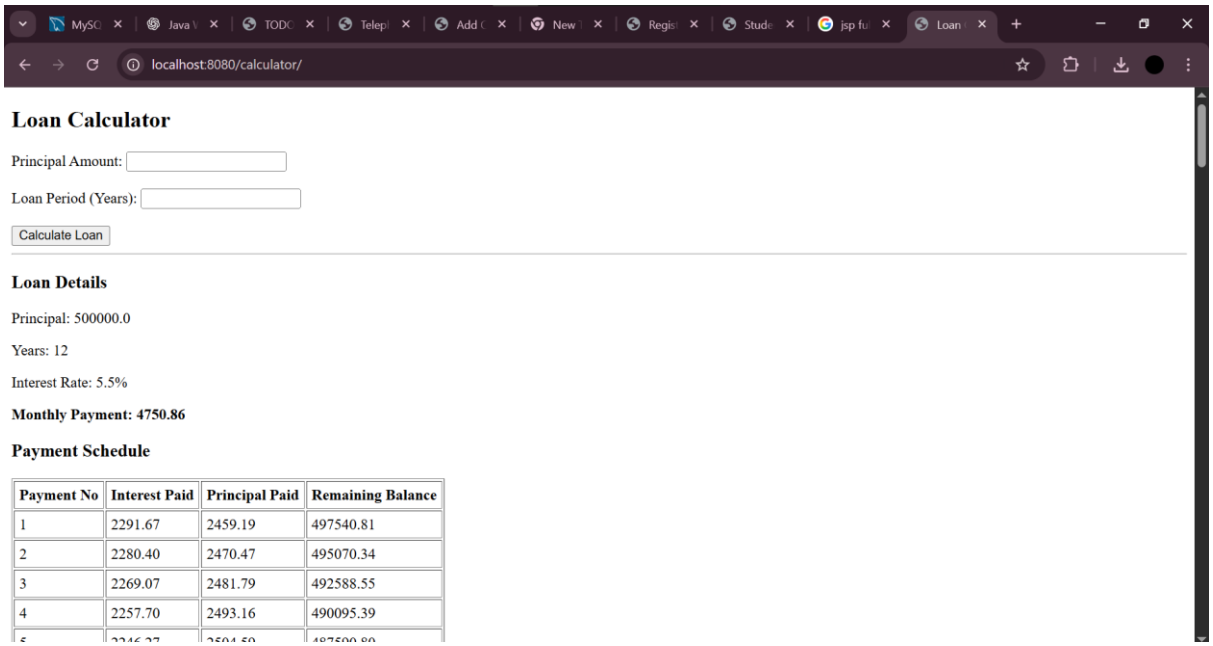
```
<%@ page language="java" %>
<!DOCTYPE html>
<html>
<head>
   <title>Loan Calculator</title>
</head>
<body>

<h2>Loan Calculator</h2>

<form method="post">
   Principal Amount:
   <input type="number" name="principal" required><br><br>

   Loan Period (Years):
   <input type="number" name="years" required><br><br>

   <input type="submit" value="Calculate Loan">
</form>

<%
```

```
if (request.getParameter("principal") != null) {

    double principal = Double.parseDouble(request.getParameter("principal"));
    int years = Integer.parseInt(request.getParameter("years"));

    double annualRate = 0;

    if (years >= 1 && years <= 7)
        annualRate = 5.35;
    else if (years >= 8 && years <= 15)
        annualRate = 5.5;
    else if (years >= 16 && years <= 30)
        annualRate = 5.75;

    double monthlyRate = annualRate / 12 / 100;
    int totalMonths = years * 12;

    double monthlyPayment =
        (principal * monthlyRate) /
        (1 - Math.pow(1 + monthlyRate, -totalMonths));

%>

<hr>
<h3>Loan Details</h3>
<p>Principal: <%= principal %></p>
<p>Years: <%= years %></p>
<p>Interest Rate: <%= annualRate %>%</p>
<p><b>Monthly Payment: <%= String.format("%.2f", monthlyPayment) %></b></p>
```

```
<h3>Payment Schedule</h3>

<table border="1" cellpadding="5">
<tr>
    <th>Payment No</th>
    <th>Interest Paid</th>
    <th>Principal Paid</th>
    <th>Remaining Balance</th>
</tr>

<%
double balance = principal;

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

    double interest = balance * monthlyRate;
    double principalPaid = monthlyPayment - interest;
    balance -= principalPaid;

    if (balance < 0) balance = 0;
%>

<tr>
    <td><%= i %></td>
    <td><%= String.format("%.2f", interest) %></td>
    <td><%= String.format("%.2f", principalPaid) %></td>
    <td><%= String.format("%.2f", balance) %></td>
</tr>

<%
```

```
}
}
%>

</table>

</body>
</html>
```

Output:



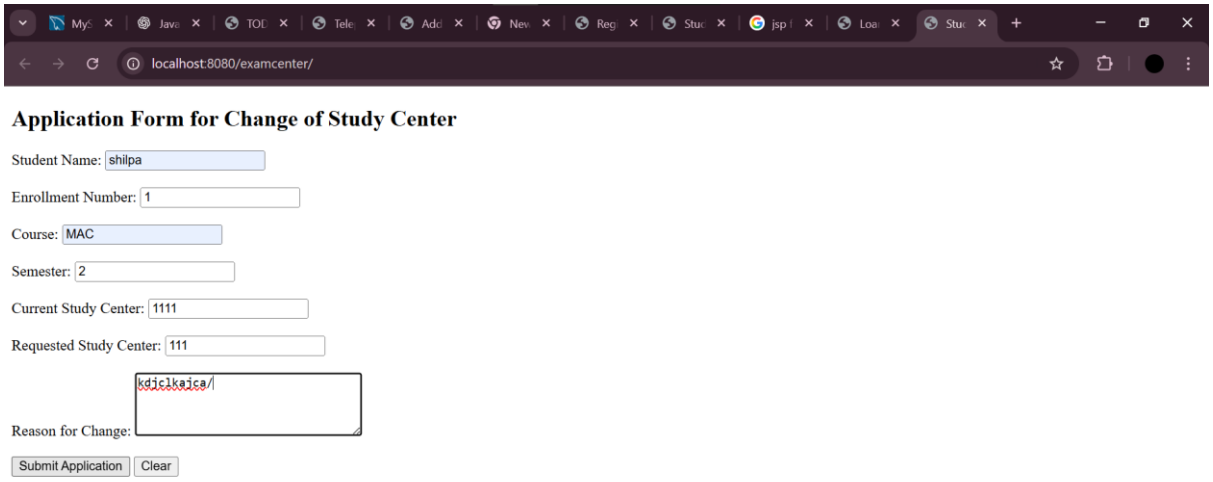
e. Write a program using JSP that displays a webpage consisting an Application form for change of Study Center which can be filled by any student who wants to change his/ her study center. Make necessary assumptions.

Code:

```
<%@ page language="java" %>
<!DOCTYPE html>
<html>
<head>
    <title>Study Center Change Application</title>
</head>
<body>

<h2>Application Form for Change of Study Center</h2>

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

    Student Name:
    <input type="text" name="name" required>
    <br><br>

    Enrollment Number:
    <input type="text" name="enroll" required>
    <br><br>

    Course:
    <input type="text" name="course" required>
    <br><br>

    Semester:
```

```
    <input type="number" name="semester" required>
    <br><br>

    Current Study Center:
    <input type="text" name="currentCenter" required>
    <br><br>

    Requested Study Center:
    <input type="text" name="newCenter" required>
    <br><br>

    Reason for Change:
    <textarea name="reason" rows="4" cols="30" required></textarea>
    <br><br>

    <input type="submit" value="Submit Application">
    <input type="reset" value="Clear">

</form>

</body>
</html>
```

Output:


localhost:8080/examcenter/
Application Form for Change of Study Center
Student Name: shilpa
Enrollment Number: 1
Course: MAC
Semester: 2
Current Study Center: 1111
Requested Study Center: 111
Reason for Change: kdjclkajca/
Submit Application
Clear

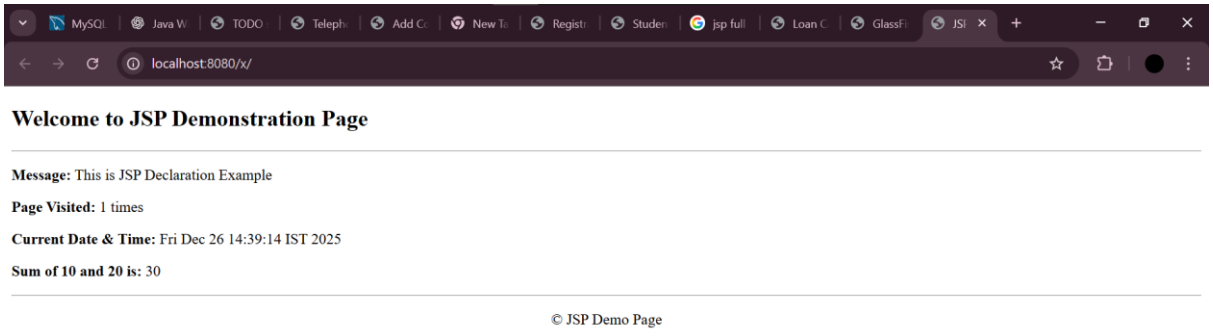
f. Write a JSP program that demonstrates the use of JSP declaration, scriptlet, directives, expression, header and footer.

Code:

```
<%@ page language="java" contentType="text/html" pageEncoding="UTF-8" %>
<%@ page import="java.util.Date" %>

<!DOCTYPE html>
<html>
<head>
   <title>JSP Elements Demo</title>
</head>

<body>

<!-- ================= HEADER ================= -->
<h2>Welcome to JSP Demonstration Page</h2>
<hr>

<!-- ================= DECLARATION ================= -->
<%!
   int count = 0;

   String getMessage() {
      return "This is JSP Declaration Example";
   }
%>

<!-- ================= SCRIPTLET ================= -->
<%
```

```
    count++;
    Date currentDate = new Date();
%>

<!-- ================= EXPRESSION ================= -->
<p><b>Message:</b> <%= getMessage() %></p>
<p><b>Page Visited:</b> <%= count %> times</p>
<p><b>Current Date & Time:</b> <%= currentDate %></p>

<!-- ================= SCRIPTLET LOGIC ================= -->
<%
    int a = 10, b = 20;
    int sum = a + b;
%>

<p><b>Sum of <%= a %> and <%= b %> is:</b> <%= sum %></p>

<!-- ================= FOOTER ================= -->
<hr>
<p style="text-align:center">
    &copy; JSP Demo Page
</p>

</body>
</html>
```

Output:


localhost:8080/x/
Welcome to JSP Demonstration Page
Message: This is JSP Declaration Example
Page Visited: 1 times
Current Date & Time: Fri Dec 26 14:39:14 IST 2025
Sum of 10 and 20 is: 30
© JSP Demo Page

7. Assignment based Spring Framework

a. Write a program to print “Hello World” using spring framework.

Code (HelloWorld.java):

```
public class HelloWorld {
   public void printMessage() {
      System.out.println("Hello World");
   }
}
```

Code (MainApp.java):

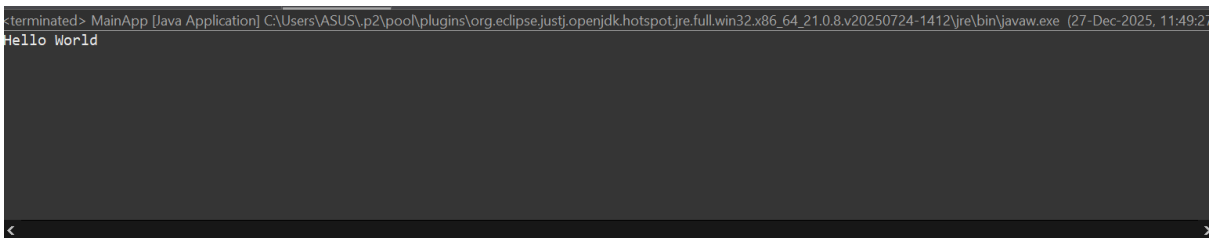
```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

      ApplicationContext context =
         new ClassPathXmlApplicationContext("applicationContext.xml");

      HelloWorld obj = (HelloWorld) context.getBean("helloBean");
      obj.printMessage();
   }
}
```

Code (applicationContext.xml):

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">
```

```
    <bean id="helloBean" class="HelloWorld"/>
</beans>
```

Output:



b. Write a program to demonstrate dependency injection via setter method.

Code (Address.java):

```
public class Address {

    private String city;

    public void setCity(String city) {
        this.city = city;
    }

    public String getCity() {
        return city;
    }
}
```

Code (Student.java):

```
public class Student {

    private Address address;

    // Setter method (Dependency Injection)
    public void setAddress(Address address) {
        this.address = address;
    }

    public void display() {
        System.out.println("Student City: " + address.getCity());
    }
}
```

Code (applicationContext.xml):

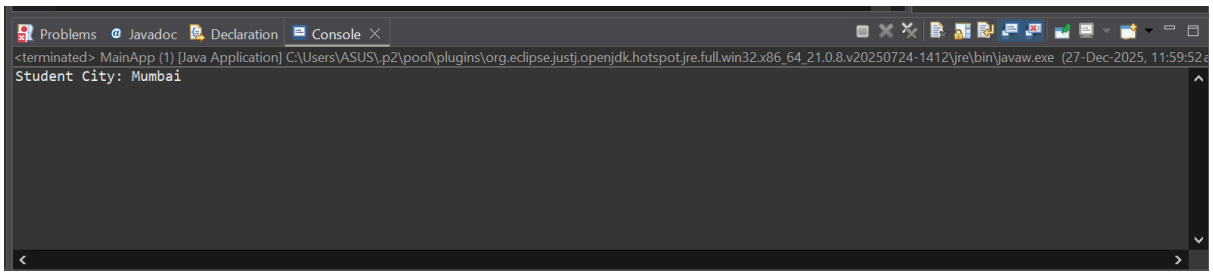
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Address Bean -->
    <bean id="addressBean" class="Address">
        <property name="city" value="Mumbai"/>
    </bean>

    <!-- Student Bean with Setter Injection -->
    <bean id="studentBean" class="Student">
        <property name="address" ref="addressBean"/>
    </bean>

</beans>
```

Code (MainApp.java):

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        Student student = (Student) context.getBean("studentBean");
```

```
        student.display();
    }
}
```

Output:



c. Write a program to demonstrate dependency injection via Constructor.

Code (Address.java):

```
public class Address {

    private String city;

    public Address(String city) {
        this.city = city;
    }

    public String getCity() {
        return city;
    }
}
```

Code(Student.java):

```
public class Student {

    private Address address;

    // Constructor Injection
    public Student(Address address) {
        this.address = address;
    }

    public void display() {
        System.out.println("Student City: " + address.getCity());
    }
}
```

Code(applicationContext.xml):

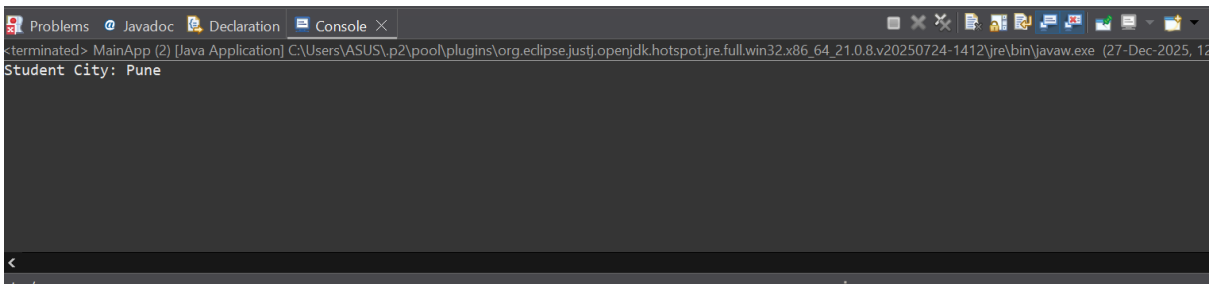
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Address Bean -->
    <bean id="addressBean" class="Address">
        <constructor-arg value="Pune"/>
    </bean>

    <!-- Student Bean using Constructor Injection -->
    <bean id="studentBean" class="Student">
        <constructor-arg ref="addressBean"/>
    </bean>

</beans>
```

Code (MainApp.java):

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        Student student = (Student) context.getBean("studentBean");
```

```
        student.display();
    }
}
```

Output:



d. Write a program to demonstrate Autowiring.

Code (Address.java):

```
public class Address {

    private String city;

    public void setCity(String city) {
        this.city = city;
    }

    public String getCity() {
        return city;
    }
}
```

Code (Student.java):

```
public class Student {

    private Address address;

    // Setter method
    public void setAddress(Address address) {
        this.address = address;
    }

    public void display() {
        System.out.println("Student City: " + address.getCity());
    }
}
```

Code (applicationContext.xml):

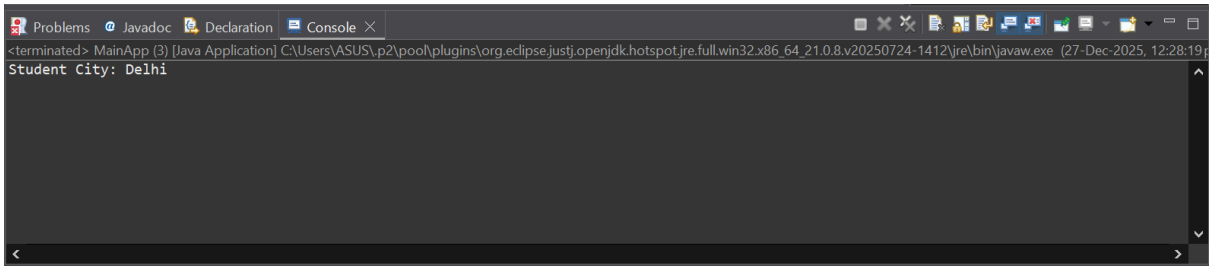
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- Address Bean -->
    <bean id="addressBean" class="Address">
        <property name="city" value="Delhi"/>
    </bean>

    <!-- Student Bean with Autowiring -->
    <bean id="studentBean" class="Student" autowire="byType"/>

</beans>
```

Code (MainApp.java):

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        Student student = (Student) context.getBean("studentBean");
        student.display();
    }
```

}

Output:


Problems
Javadoc
Declaration
Console
<terminated> MainApp (3) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\jre\bin\javaw.exe (27-Dec-2025, 12:28:19
Student City: Delhi

8. Assignment based Aspect Oriented Programming

a. Write a program to demonstrate Spring AOP – before advice.

Student.java:

```
public class Student {

    public void study() {
        System.out.println("Student is studying...");
    }
}
```

LoggingAspect.java:

```
public class LoggingAspect {

    public void beforeAdvice() {
        System.out.println("Before Advice: Logging before method execution");
    }
}
```

applicationContext.xml:

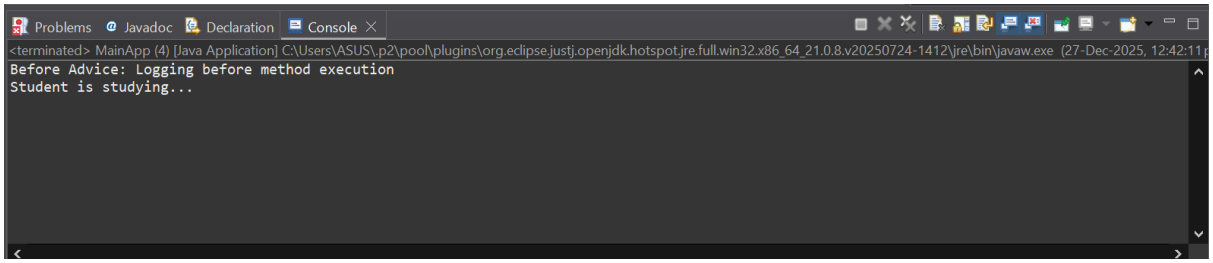
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!-- Target Bean -->
```

```
   <bean id="studentBean" class="Student"/>

   <!-- Aspect Bean -->
   <bean id="loggingAspect" class="LoggingAspect"/>

   <!-- AOP Configuration -->
   <aop:config>
      <aop:aspect ref="loggingAspect">
         <aop:before method="beforeAdvice"
                  pointcut="execution(* Student.study(..))"/>
      </aop:aspect>
   </aop:config>

</beans>
```

MainApp.java:

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

      ApplicationContext context =
         new ClassPathXmlApplicationContext("applicationContext.xml");

      Student student = (Student) context.getBean("studentBean");
      student.study();
   }
}
```

Output:


Problems
Javadoc
Declaration
Console
<terminated> MainApp (4) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\jre\bin\javaw.exe (27-Dec-2025, 12:42:11
Before Advice: Logging before method execution
Student is studying...

b. Write a program to demonstrate Spring AOP – after advice.

Student.java:

```
public class Student {

    public void study() {
        System.out.println("Student is studying...");
    }
}
```

LoggingAspect.java:

```
public class LoggingAspect {

    public void afterAdvice() {
        System.out.println("After Advice: Logging after method execution");
    }
}
```

applicationContext.xml:

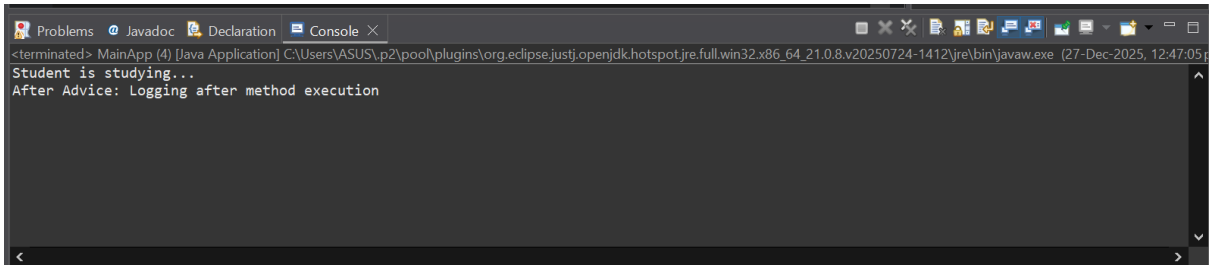
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!-- Target Bean -->
```

```
    <bean id="studentBean" class="Student"/>

    <!-- Aspect Bean -->
    <bean id="loggingAspect" class="LoggingAspect"/>

    <!-- AOP Configuration -->
    <aop:config>
        <aop:aspect ref="loggingAspect">
            <aop:after method="afterAdvice"
                    pointcut="execution(* Student.study(..))"/>
        </aop:aspect>
    </aop:config>

</beans>
```

MainApp.java:

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        Student student = (Student) context.getBean("studentBean");
        student.study();
    }
}
```

Output:


Problems
Javadoc
Declaration
Console
<terminated> MainApp (4) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\jre\bin\javaw.exe (27-Dec-2025, 12:47:05
Student is studying...
After Advice: Logging after method execution

c. Write a program to demonstrate Spring AOP – around advice.

Student.java:

```
public class Student {

    public void study() {
        System.out.println("Student is studying...");
    }
}
```

LoggingAspect.java:

```
import org.aspectj.lang.ProceedingJoinPoint;

public class LoggingAspect {

    public Object aroundAdvice(ProceedingJoinPoint pjp) throws Throwable {

        System.out.println("Around Advice: Before method execution");

        Object result = pjp.proceed();   // calls target method

        System.out.println("Around Advice: After method execution");

        return result;
    }
}
```

applicationContext.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
```

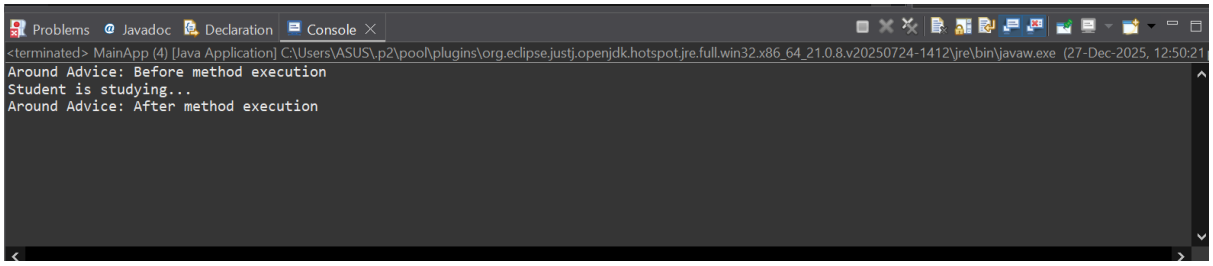
```
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd
      http://www.springframework.org/schema/aop
      http://www.springframework.org/schema/aop/spring-aop.xsd">

   <!-- Target Bean -->
   <bean id="studentBean" class="Student"/>

   <!-- Aspect Bean -->
   <bean id="loggingAspect" class="LoggingAspect"/>

   <!-- AOP Configuration -->
   <aop:config>
      <aop:aspect ref="loggingAspect">
         <aop:around method="aroundAdvice"
                  pointcut="execution(* Student.study(..))"/>
      </aop:aspect>
   </aop:config>

</beans>
```

MainApp.java:

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

      ApplicationContext context =
         new ClassPathXmlApplicationContext("applicationContext.xml");
```

```
        Student student = (Student) context.getBean("studentBean");
        student.study();
    }
}
```

Output:



d. Write a program to demonstrate Spring AOP – after returning advice.

Student.java:

```
public class Student {

    public String study() {
        System.out.println("Student is studying...");
        return "Study Completed";
    }
}
```

LoggingAspect.java:

```
public class LoggingAspect {

    public void afterReturningAdvice(Object result) {
        System.out.println("After Returning Advice: Method returned -> " + result);
    }
}
```

applicationContext.xml:

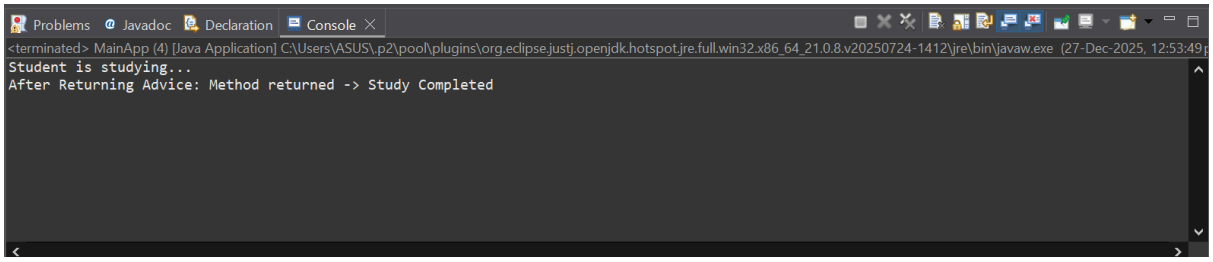
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!-- Target Bean -->
    <bean id="studentBean" class="Student"/>
```

```
    <!-- Aspect Bean -->
    <bean id="loggingAspect" class="LoggingAspect"/>

    <!-- AOP Configuration -->
    <aop:config>
        <aop:aspect ref="loggingAspect">
            <aop:after-returning
                method="afterReturningAdvice"
                pointcut="execution(* Student.study(..))"
                returning="result"/>
        </aop:aspect>
    </aop:config>

</beans>
```

MainApp.java:

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        Student student = (Student) context.getBean("studentBean");
        student.study();
    }
}
```

Output:


Problems
Javadoc
Declaration
Console
<terminated> MainApp (4) [Java Application] C:\Users\ASUS\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.8.v20250724-1412\jre\bin\javaw.exe (27-Dec-2025, 12:53:49
Student is studying...
After Returning Advice: Method returned -> Study Completed

e. Write a program to demonstrate Spring AOP – after throwing advice.

Student.java:

```
public class Student {


   public void study() {
      System.out.println("Student is studying...");
      throw new RuntimeException("Study interrupted due to error");
   }
}
```

LoggingAspect.java:

```
public class LoggingAspect {


   public void afterThrowingAdvice(Exception ex) {
      System.out.println("After Throwing Advice: Exception occurred -> " +
ex.getMessage());
   }
}
```

applicationContext.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:aop="http://www.springframework.org/schema/aop"
      xsi:schemaLocation="
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd
      http://www.springframework.org/schema/aop
      http://www.springframework.org/schema/aop/spring-aop.xsd">


   <!-- Target Bean -->
   <bean id="studentBean" class="Student"/>
```

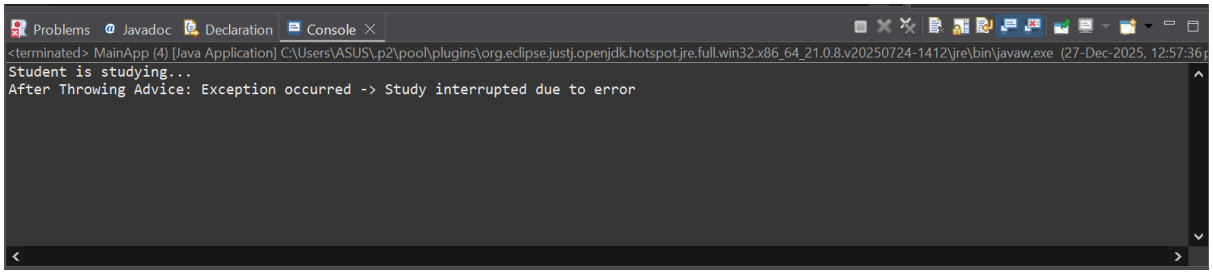
```
    <!-- Aspect Bean -->
    <bean id="loggingAspect" class="LoggingAspect"/>

    <!-- AOP Configuration -->
    <aop:config>
        <aop:aspect ref="loggingAspect">
            <aop:after-throwing
                method="afterThrowingAdvice"
                pointcut="execution(* Student.study(..))"
                throwing="ex"/>
        </aop:aspect>
    </aop:config>

</beans>
```

MainApp.java:

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        Student student = (Student) context.getBean("studentBean");

        try {
            student.study();
```

```
        } catch (Exception e) {
            // Exception handled in main
        }
    }
}
```

Output:



f. Write a program to demonstrate Spring AOP – pointcuts.

Student.java:

```
public class Student {

    public void study() {
        System.out.println("Student is studying...");
    }

    public void attendExam() {
        System.out.println("Student is attending exam...");
    }
}
```

LoggingAspect.java:

```
public class LoggingAspect {

    public void logAdvice() {
        System.out.println("Pointcut Advice: Method execution intercepted");
    }
}
```

applicationContext.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/aop
       http://www.springframework.org/schema/aop/spring-aop.xsd">
```

```
    <!-- Target Bean -->
    <bean id="studentBean" class="Student"/>

    <!-- Aspect Bean -->
    <bean id="loggingAspect" class="LoggingAspect"/>

    <!-- AOP Configuration with Pointcut -->
    <aop:config>

        <!-- Pointcut definition -->
        <aop:pointcut id="studentMethods"
                expression="execution(* Student.*(..))"/>

        <aop:aspect ref="loggingAspect">
            <aop:before method="logAdvice"
                    pointcut-ref="studentMethods"/>
        </aop:aspect>

    </aop:config>

</beans>
```

MainApp.java:

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

        ApplicationContext context =
            new ClassPathXmlApplicationContext("applicationContext.xml");
```

```
        Student student = (Student) context.getBean("studentBean");

        student.study();
        student.attendExam();
    }
}
```

Output: