Sem1Practicle - mca note for viva

Masters of Computer Applications (University of Mumbai)

JAVA

Practical no 1. -- Java Generic:

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

```
package practical1;


class Demo<T>
{
T d;
public Demo(T data) {
this.d =data;
}
public T getData() {
return d;
}
}
class demonstrateaGenericClass{
public static void main (String[] args)
{
Demo<Integer> d1 = new Demo<>(100);
Demo<String> d2 = new Demo<>("This is string");
System.out.println("Displaying Integer data "+d1.getData());
System.out.println("Displaying String data "+d2.getData());
}
}
```

Notes: no need of public class

2 : Write a program to demonstrate a Generic method:

```
package practical1;

public class todemonstrateaGenericmethod {



public static void main (String[] args)
{ EqualityClass ec = new EqualityClass();
ec.<Integer>checking(1,2);
ec.<Integer>checking(2,2);
ec.<String>checking("Hello 100","Hello 100");
ec.<String>checking("Hi","Bye");
}
}
class EqualityClass{
public <T> void checking(T d1, T d2) {
if (d1.equals(d2)) {
System.out.println(d1+" and "+d2+" are equal");
}else {
System.out.println(d1+" and "+d2+" are not equal");
}
}
}
```

Note:

1.Create java project

2.class and there will be public class

3 .a: Write a program to implement Wildcard:

```
package practical1;
import java.util.*;
public class implementwilcard {


	public static void display(List<? extends Number> list) {
	System.out.print(list);
	}
	public static void main(String args[]) {
	List<Integer> list1= Arrays.asList(4,5,6,7);
	display(list1);
	}
	}	}
```

3.b.A generic method which prints elements of the list of type number or Integer using Lower Bounded Wildcards.

```
package practical1;
import java.util.*;

	public class Unbounded {
	private static void display(List<?> list)
	{
	System.out.println(list);
	}
	public static void main(String[] args)
	{
	List<Integer> list1= Arrays.asList(1,2,3);
	List<Double> list2=Arrays.asList(1.1,2.2,3.3);
	display(list1);
	display(list2);
	}
	}
```

3.c.A generic method which prints items in the list using Unbounded Wildcards.

```
package practical1;
import java.util.*;

	public class Unbounded {
	private static void display(List<?> list)
	{
	System.out.println(list);
	}
	public static void main(String[] args)
	{
	List<Integer> list1= Arrays.asList(1,2,3);
	List<Double> list2=Arrays.asList(1.1,2.2,3.3);
	display(list1);
	display(list2);
	}
		}
```

Practical no 2. -- List interface:

Write a Java program to create a List containing a list of items of type String and use for –each loop to print the items of the list.

Ans:

```
package practicle2;
import java.util.*;

        public class ListForEach {
        public static void main(String args[]) {
        List<String> l1 = new ArrayList<>();
        l1.add("1st element");
        l1.add("2nd element");
        l1.add("3rd element");
        l1.add("4th element");
        l1.add("5th element");
        for(String str : l1) {
        System.out.println(str+" ");
        }
        }
        }
```

Question 2 : Write a Java program to create a List containing a list of items and use Listiterator interface to print items present in the list. Also print the list in reverse/backward direction.

Answer:

```
package practicle2;
import java.util.*;
public class IteratorExample {

        public static void main(String args[]) {
        List<Integer> l1 = new ArrayList<>();
        l1.add(1);
        l1.add(2);
        l1.add(3);
        ListIterator<Integer> L = l1.listIterator();
        System.out.println("Traversing in Forward direction");
        while(L.hasNext()) {
        System.out.println(L.next());
        }
        System.out.println("Traversing in reverse direction");
        while(L.hasPrevious()) {
        System.out.println(L.previous());
        }
        }
        }
```

Practical no 3. -- Set interface:

Question 1 : Write a Java program to create a Set containing a list of items of type String and print the items in the list using the Iterator interface. Also print the set elements in reverse/ backward direction.

Ans:

```
package practicle3;
import java.util.*;

public class SetIterator {

	public static void main(String[]arg) {



	Set<String> s1 = new TreeSet<>();
	s1.add("C++");
	s1.add("Java");
	s1.add("C#");
	Iterator<String> itr = s1.iterator();
	System.out.println("Traverse in Forward Direction");
	while(itr.hasNext()) {
	System.out.println(itr.next());
	}
	System.out.println("Traverse in Reverse Direction");
	List<String> l1 = new ArrayList<String>(s1);
	Collections.reverse(l1);
	for(String language : l1) {
	System.out.println(language);
	}
}
}
```

Question 2 : 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 into another set.

c.Remove items from the set.

d.Search the specified items in the set.

Ans:

```
package practicle3;
import java.util.*;

public class SetOperation {

        public static void main(String[]arg) {



        Set<Integer> id= new TreeSet<>();
        id.add(1);
        id.add(9);
        id.add(3);
        System.out.println("Items in 1st Set:" + id); //add operation
        Set<Integer> id2 = new TreeSet<>();
        id2.add(12);
        id2.add(23);
        System.out.println("Items in 2nd Set:" + id2); //insert Operation
        System.out.println("Inserting items of first set into another:");
        id.addAll(id2);
        System.out.println(id);
        if(id.contains(9)) { id.remove(9); }
        System.out.println("after deletion of item in set:"); //remove operation
        System.out.println(id);

        int searchItem = 12;
  if (id.contains(searchItem)) {
```

```
        System.out.println("Item " + searchItem + " found in the set.");
    } else {
        System.out.println("Item " + searchItem + " not found in the set.");
    }


            }
}
```

Practical no 4. -- Map interface:

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 to another map.

f) Print all keys and values of the map. (Also separate keys and Values)

Ans:

```
package practicle4;
import java.util.*;
public class MapOperation {

        public static void main(String[]arg) {

        Map<Integer,String> map=new HashMap<Integer, String>();
        map.put(1, "Sandeep M");
        map.put(2, "Afsa S");
```

```
        map.put(3, "Sandhya D");
        map.forEach((k,v) -> System.out.println(k+" "+v) );
        //Checked for the existence of the key and removed that element.
        if(map.containsKey(2))
        { map.remove(2); }
        System.out.println("After removing element with key as 2:");
        map.forEach((k,v) -> System.out.println(k+" "+v));
        //Fetched value using key
        String val=map.get(1);
        System.out.println("Name of Student with Roll no 1 is:"+val);

        Map<Integer,String> map2=new HashMap<Integer, String>();
        map2.put(4, "Janki M");
        map2.put(5, "Seema E");
        //Adding elements from one map into another
        map.putAll(map2);
        System.out.println("After adding one map elements into another.");
        map.forEach((k,v) -> System.out.println(k+" "+v));
        // using keySet() for iteration over keys
        for (Integer key : map.keySet())
        System.out.println("key: " + key);
        // using values() for iteration over values
        for (String value : map.values())
        System.out.println("value: " + value);
 }
}
```

Practical no 5. – Lambda Expression:

1. **Write a Java program using Lambda Expression to print ”Hello World”.**

Ans:

```
package practicle5;

interface Greetings {
        public void sayHelloWorld();
        }
public class LambdaExpressionDemo {
        public static void main(String[] args) {
        System.out.println("Lambda Expression Demonstration");
        Greetings greet = () -> System.out.println("Hello World");
        greet.sayHelloWorld();
        }
        }
```

2. Write a Java program using Lambda Expression with single parameters. (single parameter in method)

Ans:

```
package practicle5;

interface Greeting {
        public void sayHelloWorld(String name);
        }
public class LambdaExpressionSingleParameter {
        public static void main(String[] args)
        {
```

```
		System.out.println("Lambda Expression Demonstration for Single parameter");
		Greeting greet = (name) -> System.out.println("Hello , My name is "+name);
		greet.sayHelloWorld("Gaurav Singh");
		}

}
```

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

Ans:

```
package practicle5;

	interface Arithmetic {
		public int add(int a, int b);
		}
	public class LambdaExpressionMultipleParameter {
		public static void main(String[] args) {
		Arithmetic arth = (a, b) -> a + b;
		int sumofnumbers;
		sumofnumbers = arth.add(5, 6);
		System.out.println("The sum of 2 numbers are:" + sumofnumbers);
		}
		}
```

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

a. Convert Fahrenheit to Celsius

b. Convert Kilometers to Miles.

```
package practicle5;


    interface TemperatureInterface {
        public void fahrenheitTocelcius(double fahrenheit);
        }
        interface DistanceInterface {
        public void kilometersTometers(double kilometers);
        }
        public class LambdaExpressionToConvert {
        public static void main(String[] args) {
        TemperatureInterface temp =

        (fahrenheit) -> System.out.println((fahrenheit - 32) * (5.0 / 9.0));

        temp.fahrenheitTocelcius(97.6);
        DistanceInterface dist =

        (kilometers) -> System.out.println(kilometers * 1000);

        dist.kilometersTometers(6.3);
        }
}
```

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

Ans:

```
package practicle5;



interface Addable{
   int add(int a,int b);
}



public class LambdaExpressionWithOrWithoutReturnKeyword {
        public static void main(String[] args) {
                // Lambda expression without return keyword.
      Addable ad1=(a,b)->(a+b);
      System.out.println(ad1.add(10,20));

      // Lambda expression with return keyword.
      Addable ad2=(int a,int b)->{
                return (a+b);
                };
      System.out.println(ad2.add(100,200));
}
}
```

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

Ans:

```
package practicle5;
```

```
public class LambdaExpressionToConcatenate2String {
        interface Buildstring {
        public String concatenate(String str1, String str2);
        }

        public static void main(String[] args) {
        Buildstring bstr=(str1,str2)->str1+str2;
        String Full_name=bstr.concatenate("Sandeep ", "Waghmare");
        System.out.println("After concatenation "+Full_name);
        }
}
```

Practicle no 5: Based On JSP

Q 1. 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%

index.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<fieldset style="width: 30%; border: 3px solid black">
Index.jsp
<legend>Loan calculator</legend>
<form action="Calcu.jsp" method="post">
<table>
<tr>
<td>Loan Amount:</td>
```

```
<td><input type="text" name="pamount" /></td>
</tr>
<tr>
<td>Years</td>
<td><input type="text" name="years" /></td>
</tr>
<tr>
<td>
<td>
<td><input type="submit" value="Calculate" /></td>
</tr>
</table>
</form>
</fieldset>
</body>
</html>
```

Calcu.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
```

```
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@include file="index.jsp"%>
<%
double amount = Double.parseDouble(request.getParameter("pamount"));
int year = Integer.parseInt(request.getParameter("years"));
double rate = 0.0;
double interest = 0.0;
if (year >= 1 || year <= 7)
{
interest = (amount * 5.35) / 100;
out.println("Interest =" + interest);
out.println("\n" + "EMI=" + (amount + interest) / (year * 12));
}
else if (year >= 8 || year <= 15)
{
interest = (amount * 5.5) / 100;
out.println("Interest =" + interest);
out.println("\n" + "EMI=" + (amount + interest) / (year * 12));
}
else if (year >= 16 || year <= 30) {
interest = (amount * 5.75) / 100;
out.println("Interest =" + interest);
out.println("\n" + "EMI=" + (amount + interest) / (year * 12));
}
```

```
else {
out.println("Enter Proper year");
}
%>
</body>
</html>
```

Create New Dynamic Web Project ⇒ Select Apache Tomcat 9 version and select folder where you kept downloaded folder of respective apache version and click finish

Create Two JSP File in web app folder

Index.jsp AND Calc.jsp

Paste only body code

Q3 Write a program to demonstrate Session tracking, create a Student directory student(name,email,password)and store all the information within a db.

Ans:

FirstCreate databse and a table:

```
CREATE DATABASE mydbsession;
use mydbsession;
CREATE TABLE student_23
  (s_name varchar(20),s_password varchar(20));
SELECT * FROM student_23;
insert into student_23 values("Rahul","Rahul@123");
SELECT * FROM student_23;
```

Create dynamic web project

Create jsp file inside web app login.jsp loginuser.jsp dashboard.jsp logout.jsp

Mysql connector jar file should be inside the classpath and the webinf ->lib folder

Tap on Project file right click-> build path -> libraries-> classpath->Add External jar then select the mysql connector jar

Copy the mysql connector jar and paste inside webinf->lib folder

Create 4 file: login.jsp // loginuser.jsp // dashboard // logout.jsp

login.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<fieldset style="width: 30%; border: 3px solid black">
 <legend>Login</legend>
 <form action="loginuser.jsp" method="post">
  <table>
   <tr>
    <td>User Name:</td>
    <td>
     <input type="text" name="name" />
    </td>
   </tr>
   <tr>
    <td>Password</td>
    <td>
     <input type="password" name="pass" />
    </td>
   </tr>
   <tr>
    <td>
     <input type="submit" value="login" />
    </td>
   </tr>
  </table>
 </form>
```

```
</fieldset>;
</body>
</html>
```

Loginuser.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%@ page import="java.sql.*"%>
<%
String l_name = request.getParameter("name");
String l_Password = request.getParameter("pass");
Class.forName("com.mysql.cj.jdbc.Driver");
Connection conn
=DriverManager.getConnection("jdbc:mysql://localhost:3306/mydbsession", "root",
"root");
String query = "select * from student_23 where s_name=? and s_password=?";
PreparedStatement ps = conn.prepareStatement (query);
ps.setString(1, l_name);
ps.setString(2, l_Password);
ResultSet rs = ps.executeQuery();
if (rs.next()) {
```

```
session.setAttribute("name", I_name);
response.sendRedirect("dashboard.jsp");
} else {
response.sendRedirect("login.jsp");
}
%>
</body>
</html>
```

dashboard.jsp::

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
if (session.getAttribute("name") != null) {
%>
<h1 style="text-align: center">
Welcome "<%= session.getAttribute("name")%>"
</h1>
<br>
<a style="display: block; text-align: center" href="logout.jsp">
```

```
Click here to Logout</a>
<%
} else {
response.sendRedirect("login.jsp");
}
%>
</body>
</html>
```

Logout.jsp::

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%
session.invalidate();
%>
</body>
</html>
```

Q3. Create two JSP Files StudentMaster.jsp and DeleteStudent.jsp inside webapp/webcontent folder:

Steps:

First create a database

CREATE DATABASE StudentMasterProgram;

USE StudentMasterProgram;

CREATE TABLE studentmaster

(RollNo INT NOT NULL, Name VARCHAR (20) NOT NULL, Semester VARCHAR (20) NOT NULL, Course VARCHAR (25), PRIMARY KEY (RollNo));

insert into studentmaster values (101, "Sandeep", "SEM 4", "BTECH");

insert into studentmaster values (102, "Suraj", "SEM 3", "MCA");

insert into studentmaster values (103, "Anshuman", "SEM 1", "BTECH");

insert into studentmaster values (104, "Pawan", "SEM 2", "MCA");

select * from studentmaster;

Create a Dynamic Web project

Create jsp file inside webapp studentmaster.jsp and deletestudent.jsp and addstudent.jsp

Mysql connector jar file and JSTL(taglib) jar file should be inside the classpath and the webinf ->lib folder

Tap on Project file right click-> build path -> libraries-> classpath->Add External jar then select the mysql connector jar and JSTL jar files(taglib jar)

Copy the mysql connector jar and jstl jar files and paste inside webinf->lib folder

1.Studentmaster.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Student Master</title>
</head>
<body>
<sql:setDataSource var="db" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/StudentMasterProgram" user="root"
password="root" />
<sql:update dataSource="${db}" var="insertRow">
UPDATE studentmaster SET Semester = "Sem 6" where RollNo=1;
</sql:update>
<sql:query dataSource="${db}" var="rs"> select * from studentmaster;
</sql:query>
<h1>Student Master Table</h1>
<form action="Deletestudent.jsp">
<table>
<tr>
<td>ID</td>
<td>Name</td>
<td>Semester</td>
<td>Course</td>
<td>Action</td>
```

```
</tr>
<c:forEach var="table" items="${rs.rows}">
<tr>
<td><c:out value="${table.rollno}"></c:out></td>
<td><c:out value="${table.name}"></c:out></td>
<td><c:out value="${table.semester}"></c:out></td>
<td><c:out value="${table.course}"></c:out></td>
<td><button type="submit" name="delete"
value="${table.rollno}">Delete</button></td>
</tr>
</c:forEach>
</table>
</form>
<h1>Add a Record to the table</h1>
<form action="Addstudent.jsp">
<table>
<tr>
<td>RollNo</td>
<td><input type="number" name="roll" /></td>
</tr>
<tr>
<td>Name</td>
<td><input type="text" name="name" /></td>
</tr>
<tr>
<td>Semester</td>
<td><input type="text" name="sem" /></td>
</tr>
<tr>
```

```
<td>Course</td>
<td><input type="text" name="course" /></td>
</tr>
<tr>
<td><input type="submit" value="Add" /></td>
</tr>
</table>
</form>
</body>
</html>
```

Addstudent.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Add Student</title>
</head>
<body>
  <sql:setDataSource var="db"
    driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/StudentMasterProgram"
    user="root" password="root" />

  <sql:update dataSource="${db}" var="insertRow">
```

```
    insert into studentmaster values (
    <%= Integer.parseInt(request.getParameter("roll")) %>,
    "<%= request.getParameter("name") %>",
    "<%= request.getParameter("sem") %>",
    "<%= request.getParameter("course") %>"
    )
  </sql:update>

  <c:redirect url="Studentmaster.jsp"></c:redirect>
</body>
</html>
```

Deletestudent.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Delete Student</title>
</head>
<body>
<sql:setDataSource var="db" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/StudentMasterProgram" user="root"
password="root" />
```

```
<sql:update dataSource="${db}" > delete from studentmaster where
Rollno=<%=Integer.parseInt(request.getParameter("delete")) %>
</sql:update>
<c:redirect url="Studentmaster.jsp"></c:redirect>
</body>
</html>
```

Q4.Write a JSP page to display the Registration form (Make your own assumptions:

Step:

First create database and table

```
CREATE DATABASE JspRegistration;
use JspRegistration;
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(255),
    last_name VARCHAR(255),
    email VARCHAR(255),
    password VARCHAR(255)
);
```

Create dynamic project

Create two jsp file register.jps and registrationprocess.jsp

Mysql connector jar file should be inside the classpath and the webinf ->lib folder

Tap on Project file right click-> build path -> libraries-> classpath->Add External jar then select the mysql connector jar

Copy the mysql connector jar and paste inside webinf->lib folder

Register.jsp

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

```
<head>
 <meta charset="UTF-8">
 <title>Registration Form</title>
</head>
<body>
 <h1>Registration Form</h1>
 <form action="Registrationprocess.jsp" method="post">
 <label for="firstName">First Name:</label>
 <input type="text" id="firstName" name="firstName" required><br>
 <label for="lastName">Last Name:</label>
 <input type="text" id="lastName" name="lastName" required><br>
 <label for="email">Email:</label>
 <input type="email" id="email" name="email" required><br>
 <label for="password">Password:</label>
 <input type="password" id="password" name="password" required><br>

 <input type="submit" value="Register">
 </form>
</body>
</html>
```

Registrationprocess.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<%@ page import="java.sql.*, java.io.*"%>
```

```
<!DOCTYPE html>
<html>
<head>
 <meta charset="UTF-8">
 <title>Registration Process</title>
</head>
<body>
 <%
 // Get user input from the registration form
 String firstName = request.getParameter("firstName");
 String lastName = request.getParameter("lastName");
 String email = request.getParameter("email");
 String password = request.getParameter("password");

 try {
 // Set up the database connection
 Class.forName("com.mysql.cj.jdbc.Driver");
 String dbURL = "jdbc:mysql://localhost:3306/JspRegistration";
 String dbUser = "root";
 String dbPassword = "root";
 Connection conn = DriverManager.getConnection(dbURL, dbUser, dbPassword);
 // Create an SQL query to insert data into a 'users' table
 String insertQuery = "INSERT INTO users (first_name, last_name, email, password) VALUES
(?, ?, ?, ?)";
 PreparedStatement preparedStatement =
conn.prepareStatement(insertQuery);
 preparedStatement.setString(1, firstName);
 preparedStatement.setString(2, lastName);
 preparedStatement.setString(3, email);
 preparedStatement.setString(4, password);
```

```
// Execute the query to insert data
int rowsInserted = preparedStatement.executeUpdate();
preparedStatement.close();
conn.close();
if (rowsInserted > 0) {
out.println(" Registration successful. Data has been inserted into the database.");
} else {
out.println("Registration failed. Please try again.");
}
} catch (Exception e) {
out.println("An error occurred: " + e.getMessage());
}
%>
</body>
</html>
```

Q4. 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.

Ans: first create database with TelephoneD

USE telephoneD

```
CREATE TABLE Contacts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    address VARCHAR(255) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    email VARCHAR(255) NOT NULL
);
```

Create a Dynamic Web project

Create jsp file inside webapp studentmaster.jsp and deletestudent.jsp and addstudent.jsp

Mysql connector jar file and JSTL(taglib) jar file should be inside the classpath and the webinf ->lib folder

Tap on Project file right click-> build path -> libraries-> classpath->Add External jar then select the mysql connector jar and JSTL jar files(taglib jar)

Copy the mysql connector jar and jstl jar files and paste inside webinf->lib folder

Contact.jsp

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

<!DOCTYPE html>

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

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

<html>

<head>

  <style>

    table, th, td {

      border: 1px solid black;

    }

  </style>

</head>

<body>

  <sql:setDataSource var="db" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/telephoneD" user="root" password="root" />


  <!-- Form for adding data -->

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

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

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


    <label for="newAddress">Address:</label>

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


    <label for="newPhone">Phone:</label>

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


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

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

```
    <button type="submit" name="add">Add Contact</button>
</form>

<!-- SQL update for adding a new row -->
<c:if test="${param.add ne null}">
    <sql:update dataSource="${db}" var="insertRow">
        INSERT INTO Contacts (name, address, phone, email) VALUES (
            '${param.newName}',
            '${param.newAddress}',
            '${param.newPhone}',
            '${param.newEmail}'
        );
    </sql:update>
</c:if>

<!-- SQL query to fetch data -->
<sql:query dataSource="${db}" var="rs">
    SELECT * FROM Contacts;
</sql:query>

<!-- Displaying the table -->
<form action="DeleteContact.jsp">
    <table>
        <tr>
            <td>Name</td>
            <td>Address</td>
            <td>Phone</td>
            <td>Email</td>
```

```
        </tr>
        <c:forEach var="contact" items="${rs.rows}">
          <tr>
            <td><c:out value="${contact.name}"></c:out></td>
            <td><c:out value="${contact.address}"></c:out></td>
            <td><c:out value="${contact.phone}"></c:out></td>
            <td><c:out value="${contact.email}"></c:out></td>
            <td><button type="submit" name="delete"
value="${contact.id}">Delete</button></td>
          </tr>
        </c:forEach>
      </table>
    </form>
</body>
</html>
```

DeleteContact.jsp:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ page import="java.sql.*" %>

<sql:setDataSource var="db" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/telephoneD" user="root" password="root" />

<c:if test="${not empty param.delete}">
```

```
    <c:set var="contactId" value="${param.delete}" />


        <%-- Use PreparedStatement to prevent SQL injection --%>
        <sql:update dataSource="${db}">
            DELETE FROM Contacts WHERE id = ?
            <sql:param value="${contactId}" />
        </sql:update>
        <c:redirect url="Contact.jsp"></c:redirect>

    <c:catch var="sqlException">
        <p>Error deleting contact: ${sqlException.message}</p>
    </c:catch>
</c:if>
```

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

Ans:

Create a dynamic web project

Create 3 jsp file inside web app

Header.jsp footer.jsp and index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>JSP Example</title>
</head>
<body>

<%@ include file="header.jsp" %>
<br/>
<%!
String name="Suraj";
%>

Name:<%=name %>
<br/>

<%
out.println("MY NAME IS "+name);
%>

<%@ include file="footer.jsp" %>

</body>
</html>
```

Header.jsp

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

```
    <meta charset="UTF-8">
    <title>Header</title>
</head>
<body>

<!-- JSP directive -->
<%@ page import="java.util.Date" %>



<h1>Welcome to our JSP Example!</h1>
<h2>THIS IS HEADER</h2>


<!-- JSP scriptlet -->
<%
    Date currentDate = new Date();
    out.println("Current Date and Time is "+currentDate);
%>
</body>
</html>
```

Footer.jsp

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

<!-- JSP expression -->
<p> <%= "This is the footer." %></p>

</body>
</html>
```

Q6.Write a program using JSP that displays a webpage consisting 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

First create a database

```
CREATE DATABASE StudyCenterJsp;

use StudyCenterJsp;

CREATE TABLE StudentTable (
    studentID INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    studyCenter VARCHAR(100)
);

INSERT INTO StudentTable VALUES (4,'John Doe', 'Computer Science');
INSERT INTO StudentTable VALUES (5,'Jane Smith', 'Mathematics');


select * from StudentTable;
```

Create a Dynamic Web project

Create jsp file inside webapp StudyCenter.jsp and updateStudyCenter.jsp

Mysql connector jar file and JSTL(taglib) jar file should be inside the classpath and the webinf ->lib folder

Tap on Project file right click-> build path -> libraries-> classpath->Add External jar then select the mysql connector jar and JSTL jar files(taglib jar)

Copy the mysql connector jar and jstl jar files and paste inside webinf->lib folder

StudyCenter.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ page import="java.sql.*" %>

<html>
<head>
   <title>Display and Update Student Data</title>
</head>
<body>
   <h2>Student Data</h2>

   <table border="1">
      <tr>
         <th>Student ID</th>
         <th>Name</th>
         <th>Study Center</th>
      </tr>

      <sql:setDataSource var="db" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/StudyCenterJsp" user="root" password="root" />
      <sql:query dataSource="${db}" var="result">
         SELECT * FROM StudentTable;
      </sql:query>

      <c:forEach var="row" items="${result.rows}">
```

```
            <td><c:out value="${row.studentID}" /></td>
            <td><c:out value="${row.name}" /></td>
            <td><c:out value="${row.studyCenter}" /></td>

        </c:forEach>
    </table>

    <hr/>

    <h2>Update Study Center</h2>
    <form action="updateStudyCenter.jsp" method="post">
        <label for="updateStudentID">Student ID:</label>
        <input type="text" id="updateStudentID" name="studentID" placeholder="Enter Student ID"
required />

        <label for="updateNewStudyCenter">New Study Center:</label>
        <input type="text" id="updateNewStudyCenter" name="newStudyCenter" placeholder="Enter
New Study Center" required />

        <input type="submit" value="Update Study Center" />
    </form>

    <%-- Display Update Message --%>
    <c:if test="${param.updateMessage ne null}">
        <p>${param.updateMessage}</p>
    </c:if>

    <%@ include file="updateStudyCenter.jsp" %>
</body>
</html>
```

updateStudyCenter.jsp

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

<%
   String studentID = request.getParameter("studentID");
   String newStudyCenter = request.getParameter("newStudyCenter");

   if (studentID != null && newStudyCenter != null) {
      try {
         Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/StudyCenterJsp", "root",
"root");
         String updateQuery = "UPDATE StudentTable SET studyCenter=? WHERE
studentID=?";
         try (PreparedStatement pstmt = conn.prepareStatement(updateQuery)) {
            pstmt.setString(1, newStudyCenter);
            pstmt.setString(2, studentID);
            int rowsUpdated = pstmt.executeUpdate();

            if (rowsUpdated > 0) {
               response.sendRedirect("StudyCenter.jsp?updateMessage=Study center
updated successfully for Student ID: " + studentID);
               return;
            } else {
               response.sendRedirect("StudyCenter.jsp?updateMessage=Error updating study
center for Student ID: " + studentID);
               return;
            }
         }
      } catch (SQLException e) {
```

```
            response.sendRedirect("StudyCenter.jsp?updateMessage=Error: " +
e.getMessage());
            return;
        } finally {
            // Close resources (if needed)
        }
    } else if (request.getMethod().equalsIgnoreCase("POST")) {
        response.sendRedirect("StudyCenter.jsp?updateMessage=Invalid parameters. Please
provide both Student ID and new Study Center.");
        return;
    }
%>
```

Module 3

Practical 7

Assignment based Spring Framework

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

Create a java project

Create a package inside the src

Inside src/package create the two java file HelloWorld.java and App.java

Create a config.xml file inside the src (outside of package)

Add the spring jar file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring jars and apply then apply and close

In config in bean tag always check or write package and class name//-eg-

```
<bean id="helloworld" class="spring1practicle.HelloWorld"></bean>
```

HelloWorld.java

```
package spring1practicle;

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

App.java

```
package spring1practicle;

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

public class App {

        public static void main(String[] args) {
            ApplicationContext context = new
ClassPathXmlApplicationContext("config.xml");
            HelloWorld hw = (HelloWorld) context.getBean("helloworld");
            hw.display();
        }
    }
```

Config.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">

  <!-- Your bean definitions go here -->
```

<bean id="helloworld" class="spring1practicle.HelloWorld"></bean>

</beans>

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

Create a java project

Create a package inside the src

Inside src/package create the two java file Account.java and App.java

Create a config.xml file inside the src (outside of package)

Add the spring jar file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring jars and apply then apply and close

In config in bean tag always check or write package and class name//-eg-

```
<bean id="helloworld" class="spring1practicle.HelloWorld"></bean>
```

Account.java:

package setterinjection;

```
public class Account {
private int acc;
private String accName;
public int getAcc() {
return acc;
}
public void setAcc(int acc) {
this.acc = acc;
}
public String getAccName() {
return accName;
} public void setAccName(String accName) {
this.accName = accName;
}
}
```

App.java:

```
package setterinjection;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class App {
public static void main(String[] args) {
ApplicationContext context = new ClassPathXmlApplicationContext("config.xml");
Account acc = (Account) context.getBean("acc");
System.out.println("Account No = "+ acc.getAcc());
System.out.println("Account Name = " + acc.getAccName());
}
}
```

Config.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:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
<bean id="acc" class="setterinjection.Account">
<property name="acc" value="05">
</property>
<property name="accName" value="HDFC">
</property>
</bean>
</beans>
```

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

Create a java project

Create a package inside the src

Inside src/package create the three java file Address.java Employee.java and App.java

Create a config.xml file inside the src (outside of package)

Add the spring jar file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring jars and apply then apply and close

In config in bean tag always check or write package and class name//-eg-

```
<bean id="helloworld" class="spring1practicle.HelloWorld"></bean>
```

Address.java

```
package constructorinject;
public class Address {
private String city;
private String state;
private String country;
public Address(String city, String state, String country) {
super();
this.city = city;
this.state = state;
this.country = country;
}
public String toString() {
return city +"" + state +"" + country;
}
```

}

Employee.java:

```
package constructorinject;
public class Employee {
private int id;
private String name;
private Address address;
//Aggregation
public Employee() { System.out.println("def cons");}
public Employee (int id, String name, Address address) {
super();
this.id = id;
this.name = name;
this.address = address;
}
void show(){
System.out.println(id+" "+name);
System.out.println(address.toString());
}
}
```

App.java

```
package constructorinject;
public class Employee {
private int id;
private String name;
private Address address;
//Aggregation
public Employee() { System.out.println("def cons");}
public Employee (int id, String name, Address address) {
super();
this.id = id;
this.name = name;
this.address = address;
}
void show(){
System.out.println(id+" "+name);
System.out.println(address.toString());
}
}
}
```

Config.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:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
<bean id="a1" class="constructorinject.Address">
<constructor-arg value="Mumbai" index="0"></constructor-arg>
<constructor-arg value="Maharashtra" index="2"></constructor-arg>
<constructor-arg value="India" index="1"></constructor-arg>
</bean>
<bean id="e" class="constructorinject.Employee">
<constructor-arg value="102" type="int"></constructor-arg>
<constructor-arg value="Shrikant"></constructor-arg>
<constructor-arg>
<ref bean="a1" />
</constructor-arg>
</bean>
</beans>
```

Practical 8

Assignment based Aspect Oriented Programming

1. Write a program to demonstrate Spring AOP – before advice.
2. Write a program to demonstrate Spring AOP – after advice.

Create a java project

Create a package inside the src

Inside src/package create the java file Student.java StudentMainApp.java and LoggingAspect.java

Create a config.xml file inside the src (outside of package)

Add the spring jar file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring aop jars and apply then apply and close

Student.java

```
package aop;
public class Student {
private int ID;
private String NAME;
public int getID() {
 return ID;
}
public void setID(int iD) {
ID = iD;
}
public String getNAME() {
return NAME;
}
public void setNAME(String nAME) {
NAME = nAME;
}
public void display()
{
System.out.println("ID: "+ID);
System.out.println("Name:"+NAME);
}
}
```

LoggingAspect.java

```
package aop;

public class LoggingAspect {
      //user defined methods ==> cross-cutting concern / Non-Functional
      public void beforeadvice() {
      System.out.println("I am BEFORE ADVICE");
      }
```

```
	public void afteradvice() {
	System.out.println("I am AFTER ADVICE");
	}
	}
```

StudentMain.java

```
package aop;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class StudentMainApp {
public static void main(String[] args) {
// TODO Auto-generated method stub
ApplicationContext context =new
ClassPathXmlApplicationContext("config.xml");
Student s= (Student) context.getBean("stud");
s.display();
}
}
```

Config.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:p="http://www.springframework.org/schema/p"
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 ">

<bean id="stud" class="aop.Student">
<property name="ID" value="56"></property>
<property name="NAME" value="Shweta Waghmare"></property>
</bean>
<bean id="Logbean" class="aop.LoggingAspect"></bean>
<aop:aspectj-autoproxy></aop:aspectj-autoproxy>
<aop:config>
<aop:aspect id="log" ref="Logbean">
<aop:before method="beforeadvice"
pointcut="execution(public void display())" />
<aop:after method="afteradvice"
pointcut="execution(public void display())" />
</aop:aspect>
</aop:config>
</beans>
```

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

Create a java project

Create a package inside the src

Inside src/package create the java file Student.java StudentMainApp.java and LoggingAspect.java

Create a config.xml file inside the src (outside of package)

Add the spring jar file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring aop jars and apply then apply and close

Student.java

```
package aop;
public class Student {
private int ID;
private String NAME;
public int getID() {
 return ID;
}
public void setID(int iD) {
ID = iD;
}
public String getNAME() {
return NAME;
}
public void setNAME(String nAME) {
NAME = nAME;
}
public void display()
{
System.out.println("ID: "+ID);
System.out.println("Name:"+NAME);
}
}
```

LoggingAspect.java

```
package aop;
import org.aspectj.lang.ProceedingJoinPoint;
public class LoggingAspect {
public void aroundadvice (ProceedingJoinPoint jp) throws Throwable {
System.out.println("Student data (Around Advice)");
jp.proceed();
System.out.println("Above data is of required student(Around Advice)");
}
```

```
}
```

StudentMain.java

```
package aop;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class StudentMainApp {
public static void main(String[] args) {
ApplicationContext context =new
ClassPathXmlApplicationContext("config.xml");
Student s= (Student) context.getBean("stud");
s.display();
}
}
```

Config.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:p="http://www.springframework.org/schema/p"
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 "
>

<bean id="stud" class="aop.Student">
<property name="ID" value="56"></property>
<property name="NAME" value="Shweta Waghmare"></property>
</bean>
<bean id="1" class="aop.LoggingAspect"></bean>
<aop:aspectj-autoproxy></aop:aspectj-autoproxy>
<aop:config>
<aop:aspect id="Logging Aspect" ref="1">
<aop:around method="aroundadvice" pointcut="execution(public void
display())"/>
</aop:aspect>
</aop:config>
</beans>
```

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

Create a java project

Create a package inside the src

Inside src/package create the java file Student.java StudentMainApp.java and LoggingAspect.java

Create a config.xml file inside the src (outside of package)

Add the spring jar file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring aop jars and apply then apply and close

Student.java

```
package aop;
public class Student {
private int ID;
private String NAME;
public int getID() {
 return ID;
}
public void setID(int iD) {
ID = iD;
}
public String getNAME() {
return NAME;
}
public void setNAME(String nAME) {
NAME = nAME;
}
public void display()
{
System.out.println("ID: "+ID);
System.out.println("Name:"+NAME);
}
}
```

LoggingAspect.java

```
package aop;
public class LoggingAspect {
public void afterreturnadvice(Object retval) {
System.out.println("Method got successfully executed (After
Returning(value) Advice )");
}
}
```

StudentMain.java

```
package aop;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class StudentMainApp {
public static void main(String[] args) {
ApplicationContext context =new
ClassPathXmlApplicationContext("config.xml");
Student s= (Student)context.getBean("stud");
s.display();
}
}
```

Config.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:p="http://www.springframework.org/schema/p"
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 "
>
<bean id="stud" class="aop.Student">
<property name="ID" value="56"></property>
<property name="NAME" value="Shweta Waghmare"></property>
</bean>
<bean id="1" class="aop.LoggingAspect"></bean>
<aop:aspectj-autoproxy></aop:aspectj-autoproxy><aop:config>
<aop:aspect id="LoggingAspect" ref="1">
<aop:after-returning method="afterreturnadvice" returning="retval"
pointcut="execution(public void display())"/>
</aop:aspect></aop:config>
</beans>
```

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

Create a java project

Create a package inside the src

Inside src/package create the java file Student.java StudentMainApp.java and LoggingAspect.java

Create a config.xml file inside the src (outside of package)

Add the spring jar file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring aop jars and apply then apply and close

Student.java

```
package aop;
public class Student {
private int ID;
public int getID() {
	return ID;
}
public void setID(int iD) {
	ID = iD;
}
public String getNAME() {
	return NAME;
}
public void setNAME(String nAME) {
	NAME = nAME;
}
public int getAge() {
	return age;
}
public void setAge(int age) {
	this.age = age;
}
private String NAME;
private int age;
//getter and setter
public void display() {
System.out.println("ID:" + ID);
System.out.println("Name:" + NAME);
}
public void validate(int age) {
if (age < 18) {
throw new ArithmeticException();
} else {
System.out.println("Valid age");
}
}
}
```

LoggingAspect

```
package aop;
public class LoggingAspect {
public void afterexceptionadvice(Exception error) {
System.out.println("Some Exception Occured");
System.out.println(error);
}
}
```

StudentMainApp.java

```
package aop;

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

public class StudentMainApp {
public static void main(String[] args) {
ApplicationContext context =new
ClassPathXmlApplicationContext("config.xml");

Student s=(Student) context.getBean("stud");
s.display();
System.out.println("After throwing advice:");
s.validate(24);
}
}
```

Conig.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:p="http://www.springframework.org/schema/p"
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 "
>
<bean id="stud" class="aop.Student">
<property name="ID" value="56"></property>
<property name="NAME" value="Shweta Waghmare"></property>
</bean>
<bean id="1" class="aop.LoggingAspect"></bean>
<aop:aspectj-autoproxy></aop:aspectj-autoproxy><aop:config>
<aop:aspect id="LoggingAspect" ref="1">
<aop:after-throwing method="afterexceptionadvice" throwing="error"
pointcut="execution(public void validate(*))"/>
</aop:aspect></aop:config>
</beans>
```

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

Create a java project

Create a package inside the src

Inside src/package create the java file MyService.java SMainApp.java and LoggingAspect.java

Create a config.xml file inside the src (outside of package)

Add the spring jar file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring aop jars and apply then apply and close

MyService.java

```
package aop;

public class MyService {

    public void performAction() {
        System.out.println("Executing the main action...");
    }

    public void anotherAction() {
        System.out.println("Executing another action...");
    }
}
```

LoggingAspect.java

```
package aop;


import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.After;

@Aspect
public class LoggingAspect {

    @Before("execution(* com.example.service.*.*(..))")
    public void beforeAdvice() {
        System.out.println("Before executing the method...");
    }

    @After("execution(* com.example.service.*.*(..))")
    public void afterAdvice() {
        System.out.println("After executing the method...");
    }
}
```

MainApp.java

```
package aop;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("config.xml");

        MyService myService = (MyService) context.getBean("myService");
        myService.performAction();

        context.close();
    }
}
```

Config.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">

    <bean id="myService" class="aop.MyService"/>

    <bean id="loggingAspect" class="aop.LoggingAspect"/>

    <aop:config>
    <aop:aspect ref="loggingAspect">
        <aop:pointcut id="serviceMethods" expression="execution(*
aop.MyService.*(..))"/>
        <aop:before pointcut-ref="serviceMethods" method="beforeAdvice"/>
        <aop:after pointcut-ref="serviceMethods" method="afterAdvice"/>
    </aop:aspect>
</aop:config>


</beans>
```

Practical 9

Assignment based Spring JDBC

1. Write a program to insert, update and delete records from the given table

First create a database

Create database JdbcUID;

use JdbcUID;

create table employee(id int, name varchar(20), salary varchar(10));

insert into employee values(89,"suraj",10000);

select *from employee;

Create a java project

Create a package inside the src

Inside src/package create the java file Employee.java EmployeeDao.java and App.java

Create a config.xml file inside the src (outside of package)

Add the spring JDBC jars file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring JDBC jars and apply then apply and close

Employee.java

```
package jdbc;

public class Employee {
	private int id;
	private String name;
	private float salary;

	public Employee() {}

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
```

```
	}
	public void setName(String name) {
		this.name = name;
	}
	public Employee(int id, String name, float salary) {
		super();
		this.id = id;
		this.name = name;
		this.salary = salary;
	}
	public float getSalary() {
		return salary;
	}
	public void setSalary(float salary) {
		this.salary = salary;
	}


}
```

EmployeeDao.java

```
package jdbc;

import org.springframework.jdbc.core.JdbcTemplate;

public class EmployeeDao {
   private JdbcTemplate jdbcTemplate;
    public void setJdbcTemplate (JdbcTemplate jdbcTemplate) {
      this.jdbcTemplate=jdbcTemplate;
}
 public int saveEmployee (Employee e) {
String query =
"insert into employee values('" + e.getId() + "','" + e.getName() + "','" +
e.getSalary() + "')";
return jdbcTemplate.update(query);
}
public int updateEmployee (Employee e) {
String query =
"update employee set name='" + e.getName() + "', salary='" + e.getSalary()
+ "' where id='"+ e.getId() + "' ";
return jdbcTemplate.update(query);
}
public int deleteEmployee (Employee e) {
String query =
"delete from employee where id='" + e.getId() + "' ";
return jdbcTemplate.update(query);
}
}
```

App.java

```java
package jdbc;

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

public class App {
public static void main(String[] args) {
// TODO Auto-generated method stub
ApplicationContext context= new
ClassPathXmlApplicationContext("config.xml");
EmployeeDao dao=(EmployeeDao)context.getBean("e123");
int status= dao.saveEmployee (new Employee (105,"Shweta",10000));
System.out.println(status +" Empolyee added");
Employee e=new Employee();
e.setId(109);
e.setName("Sandeep");
e.setSalary(200000);
int result= dao.saveEmployee(e);
System.out.println(result +" Empolyee added");

int status2= dao.updateEmployee(new Employee(109,"NewSandeep",5000));
System.out.println(status2 +" Empolyee updated");
//Deletion
Employee e2=new Employee();
e2.setId(546);
e2.setName("DeleteName");
e2.setSalary(80000);
dao.saveEmployee(e2);
System.out.println(dao.deleteEmployee(e2)+" Empolyee Deleted");
}
}
```

Config.xml

```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:p="http://www.springframework.org/schema/p"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

 <bean id="ds"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="com.mysql.jdbc.Driver" />
<property name="url" value="jdbc:mysql://localhost:3306/JdbcUID" />
<property name="username" value="root" />
<property name="password" value="root" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>
<bean id="e123" class="jdbc.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

Check the database after the localhost3306/

And package and class name at class=””

2. Write a program to demonstrate PreparedStatement in Spring JdbcTemplate

First create a database

Create database JdbcPreparedStatement1;

use JdbcPreparedStatement1;

create table employee(id int, name varchar(20), salary varchar(10));

insert into employee values(89,"suraj",10000);

select *from employee;

Create a java project

Create a package inside the src

Inside src/package create the java file Employee.java EmployeeDao.java and App.java

Create a config.xml file inside the src (outside of package)

Add the spring JDBC jars file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring JDBC jars and apply then apply and close

Employee.java

```
package jdbc;

public class Employee {
	private int id;
	private String name;
	private float salary;

	public Employee() {}

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public Employee(int id, String name, float salary) {
		super();
		this.id = id;
```

```
            this.name = name;
            this.salary = salary;
      }
      public float getSalary() {
            return salary;
      }
      public void setSalary(float salary) {
            this.salary = salary;
      }


}
```

EmployeeDao.java

```
package jdbc;

import java.sql.PreparedStatement;
import java.sql.SQLException;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCallback;

public class EmployeeDao {
private JdbcTemplate jdbcTemplate;
public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
this.jdbcTemplate=jdbcTemplate;
}
Boolean saveEmployeebyPrepared(Employee e) {
String query = "insert into employee values(?,?,?)";
return jdbcTemplate.execute(query, new PreparedStatementCallback<Boolean>()
{
@Override
public Boolean doInPreparedStatement (PreparedStatement ps) throws
SQLException, DataAccessException {
ps.setInt(1, e.getId());
ps.setString(2, e.getName());
ps.setFloat(3, e.getSalary());
return ps.execute();
}
});
}
}
```

App.java

```
package jdbc;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class App {
public static void main(String[] args) {
// TODO Auto-generated method stub
ApplicationContext context = new
ClassPathXmlApplicationContext("config.xml");
EmployeeDao dao = (EmployeeDao) context.getBean("e123");
```

```
if(!dao.saveEmployeebyPrepared(new Employee (108, "Amit", 35000))){
      System.out.println("Employee Stored Sucessfully using prepared
statement");
      }
else {
      System.out.println("Erorrrr!!!!!");
}

}
}
```

Config.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:p="http://www.springframework.org/schema/p"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

 <bean id="ds"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="com.mysql.jdbc.Driver" />
<property name="url"
value="jdbc:mysql://localhost:3306/JdbcPreparedStatement" />
<property name="username" value="root" />
<property name="password" value="root" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>
<bean id="e123" class="jdbc.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

3. Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface

First create a database

Create database JdbcResultSetExtractor;

use JdbcResultSetExtractor;

create table employee(id int, name varchar(20), salary varchar(10));

insert into employee values(89,"suraj",10000);

insert into employee values(117,"anshuman",20000);

insert into employee values(118,"ayush",30000);

select *from employee;

Create a java project

Create a package inside the src

Inside src/package create the java file Employee.java EmployeeDao.java and App.java

Create a config.xml file inside the src (outside of package)

Add the spring JDBC jars file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring JDBC jars and apply then apply and close

Employee.java

```
package jdbc;

public class Employee {
	private int id;
	private String name;
	private float salary;

	public Employee() {}

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
```

```
	}
	public Employee(int id, String name, float salary) {
		super();
		this.id = id;
		this.name = name;
		this.salary = salary;
	}
	public float getSalary() {
		return salary;
	}
	public void setSalary(float salary) {
		this.salary = salary;
	}


}
```

EmployeeDao.java

```
package jdbc;

import java.util.ArrayList;
import java.util.List;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCallback;
import org.springframework.jdbc.core.ResultSetExtractor;
import java.sql.ResultSet;
import java.sql.SQLException;

public class EmployeeDao {
private JdbcTemplate jdbcTemplate;
public void setJdbcTemplate (JdbcTemplate jdbcTemplate) {
this.jdbcTemplate = jdbcTemplate;
}
public List<Employee> getAllEmployees(){
return jdbcTemplate.query("select * from employee", new
ResultSetExtractor<List<Employee>>(){
@Override
public List<Employee> extractData(ResultSet rs) throws SQLException,
DataAccessException {
List<Employee> list=new ArrayList<Employee>();
while(rs.next()){
Employee e=new Employee();
e.setId(rs.getInt(1));
e.setName(rs.getString(2));
e.setSalary(rs.getInt(3));
list.add(e);
}
return list;
}
});
}
}
```

App.java

```java
package jdbc;

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

import java.util.List;
public class App {
public static void main(String[] args) {
// TODO Auto-generated method stub
ApplicationContext context = new
ClassPathXmlApplicationContext("config.xml");
EmployeeDao dao = (EmployeeDao) context.getBean("e123");
System.out.println("Employee Data : ");
List<Employee> list = dao.getAllEmployees();
for (Employee display:list) {
System.out.print(" " + display.getId());
System.out.print(" " +display.getName());
System.out.print(" " +display.getSalary());
System.out.println();
System.out.println("----------------------");
}
}
}
```

Config.xml

```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:p="http://www.springframework.org/schema/p"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

 <bean id="ds"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="com.mysql.jdbc.Driver" />
<property name="url"
value="jdbc:mysql://localhost:3306/JdbcResultSetExtractor" />
<property name="username" value="root" />
<property name="password" value="root" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>
<bean id="e123" class="jdbc.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

4. Write a program to demonstrate RowMapper interface to fetch the records from the database.

First create a database

Create database JdbcRowMapper;

use JdbcRowMapper;

create table employee(id int, name varchar(20), salary varchar(10));

insert into employee values(89,"suraj",10000);

insert into employee values(117,"anshuman",20000);

insert into employee values(118,"ayush",30000);

select *from employee;

Create a java project

Create a package inside the src

Inside src/package create the java file Employee.java EmployeeDao.java and App.java

Create a config.xml file inside the src (outside of package)

Add the spring JDBC jars file

Right click on the project -> build path-> configure build path-> libraries->classpath->addexternaljars->select all spring JDBC jars and apply then apply and close

Employee.java

```
package jdbc;

public class Employee {
	private int id;
	private String name;
	private float salary;

	public Employee() {}

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
```

```
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public Employee(int id, String name, float salary) {
		super();
		this.id = id;
		this.name = name;
		this.salary = salary;
	}
	public float getSalary() {
		return salary;
	}
	public void setSalary(float salary) {
		this.salary = salary;
	}


}
```

EmployeeDao.java

```
package jdbc;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
public class EmployeeDao {
private JdbcTemplate jdbcTemplate;
public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
this.jdbcTemplate = jdbcTemplate;
}
public List<Employee> getAllByRowmapper() {
return jdbcTemplate.query("select* from employee", new
RowMapper<Employee>() {
@Override
public Employee mapRow(ResultSet rs, int col_no) throws SQLException {
Employee e = new Employee();
e.setId(rs.getInt(1)); e.setName(rs.getString(2));
e.setSalary(rs.getFloat(3));
return e;
}
});
}
}
```

App.java

```
package jdbc;

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

```
import java.util.List;
public class App {
public static void main(String[] args) {
// TODO Auto-generated method stub
ApplicationContext context = new
ClassPathXmlApplicationContext("config.xml");
EmployeeDao dao = (EmployeeDao) context.getBean("e123");
System.out.println("Employee Data: Using Rowmapper");
List<Employee> list2 = dao.getAllByRowmapper();
for (Employee display : list2) {
System.out.print(" " + display.getId());
System.out.print(" " +display.getName());
System.out.print(" " +display.getSalary());
System.out.println();
System.out.println("--------------------");
}
}
}
```

Config.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:p="http://www.springframework.org/schema/p"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

 <bean id="ds"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="com.mysql.jdbc.Driver" />
<property name="url" value="jdbc:mysql://localhost:3306/JdbcRowMapper" />
<property name="username" value="root" />
<property name="password" value="root" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>
<bean id="e123" class="jdbc.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```