

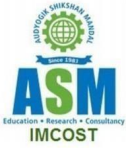
**MCA A.Y. 2021-2022**

**Journals of Advanced Java**

**ASM's Institute of Management & Computer Studies,  
Thane**



<b>Candidate Full Name</b>	Yashita Digambar Parab
<b>Roll No.</b>	MC2122087
<b>Course</b>	Masters of Computer Applications (MCA)
<b>Semester</b>	I
<b>Subject Faculty Incharge</b>	Prof. Abhay More



**Audyogik Shikshan Mandal's  
Institute of Management & Computer Studies, Thane**

## **Certificate**

This is to certify that Mr. / Ms. **Yashita Digambar Parab** Student of **MCA** Course, First year, **Semester 1**, Roll No **MC2122087** has successfully Complete the required number of practical in subject of **Advanced Java Lab** as prescribed by the **University of Mumbai** under our supervision during the academic-year 2021- 2022.

**Practical In-Charge**  
**Date:**

**Internal Examiner**  
**Date:**

**External Examiner**  
**Date:**

**Director**  
**IMCOST**

**College Seal:**

## INDEX

Sr. No.	Topic	Sign
<b>1.</b>	<b>Practical No. 1</b>	
	<b>Assignments on Java Generics</b>	
	1. Write a Java Program to demonstrate a Generic Class.	
	2. Write s Java Program to demonstrate Generic Methods.	
	3. Write a Java Program to demonstrate Wildcards in Java Generics.	
<b>2.</b>	<b>Practical No. 2</b>	
	<b>Assignments on List Interface</b>	
	1. Write a Java program to create List containing list of items of type String and use for---each loop to print the items of the list.	
	2. Write a Java program to create List containing list of items and use ListIterator interface to print items present in the list. Also print the list in reverse/ backward direction.	
<b>3.</b>	<b>Practical No. 3</b>	
	<b>Assignments on Set Interface</b>	
	1. Write a Java program to create a Set containing list of items of type String and printthe items in the list using Iterator interface. Also print the list in reverse/ backward direction.	
	2. Write a Java program using Set interface containing list of items and perform the following operations: a. Add items in theset. b. Insert items of one set in to other set. c. Remove items from theset d. Search the specified item in theset	
<b>4.</b>	<b>Practical No. 4</b>	
	<b>Assignments on Map Interface</b>	
	1. 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 themap. b. Remove items from themap	

## INDEX

	<ul style="list-style-type: none"> <li>c. Search specific key from the map</li> <li>d. Get value of the specified key</li> <li>e. Insert map elements of one map in to other map.</li> <li>f. Print all keys and values of the map.</li> </ul>	
<b>5.</b>	<b>Practical No. 5</b>	
	<b>Assignments on Lambda Expression</b>	
	1. Write a Java program using Lambda Expression to print "Hello World".	
	2. Write a Java program using Lambda Expression with single parameters.	
	3. Write a Java program using Lambda Expression with multiple parameters to add two numbers.	
	4. Write a Java program using Lambda Expression to calculate the following: <ul style="list-style-type: none"> <li>a. Convert Fahrenheit to Celcius</li> <li>b. Convert Kilometers to Miles.</li> </ul>	
	5. Write a Java program using Lambda Expression with or without return keyword.	
	6. Write a Java program using Lambda Expression to concatenate two strings.	
<b>6.</b>	<b>Practical No. 6</b>	
	<b>Assignments based on web application development using JSP</b>	
	1. 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.	
	2. Write a JSP page to display the Registration form (Make your own assumptions)	
	3. Write a JSP program to add, delete and display the records from StudentMaster (RollNo, Name, Semester, Course) table.	
	4. 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: <ul style="list-style-type: none"> <li>a. 1 to 7 year at 5.35%</li> <li>b. 8 to 15 year at 5.5%</li> <li>c. 16 to 30 year at 5.75%</li> </ul>	
	5. 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	
	6. Write a JSP program to add, delete and display the records from StudentMaster (RollNo, Name, Semester, Course) table.	

## INDEX

	7. Write a JSP program that demonstrates the use of JSPdeclaration, scriptlet, directives, expression, header and footer.	
<b>7.</b>	<b>Practical No. 7</b>	
	<b>Assignment based Spring Framework</b>	
	1. Write a program to print “Hello World” using spring framework.	
	2. Write a program to demonstrate dependency injection via setter method.	
	3. Write a program to demonstrate dependency injection via Constructor.	
<b>8.</b>	<b>Practical No. 8</b>	
	<b>Assignment based Aspect Oriented Programming</b>	
	1. Write a program to demonstrate Spring AOP – before advice.	
	2. Write a program to demonstrate Spring AOP – after advice.	
	3. Write a program to demonstrate Spring AOP – around advice.	
	4. Write a program to demonstrate Spring AOP – after returning advice.	
	5. Write a program to demonstrate Spring AOP – after throwing advice.	
	6. Write a program to demonstrate Spring AOP – pointcuts.	
<b>9.</b>	<b>Practical No. 9</b>	
	<b>Assignment based Spring JDBC</b>	
	1. Write a program to insert, update and delete records from the given table.	
	2. Write a program to demonstrate PreparedStatement in Spring JdbcTemplate	
	3. Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface	
	4. Write a program to demonstrate RowMapper interface to fetch the records from the database.	
<b>10.</b>	<b>Practical No. 10</b>	
	<b>Assignment based Spring Boot and RESTful Web Services</b>	
	1. Write a program to create a simple Spring Boot application that prints a message.	
	2. Write a program to demonstrate RESTful Web Services with spring boot.	

# **Practical No 1**

## **AIM: Assignments on Lambda Expression**

### **Description:**

Generics means parameterized types. The idea is to allow type (Integer, String, etc, and user-defined types) to be a parameter to methods, classes, and interfaces. Using Generics, it is possible to create classes that work with different data types.

The Java Generics programming is introduced in J2SE 5 to deal with type-safe objects. It makes the code stable by detecting the bugs at compile time.

Before generics, we can store any type of objects in the collection, i.e., non- generic. Now generics force the java programmer to store a specific type of objects.

### **A. Write a Java Program to demonstrate a Generic Class.**

Code:

```
package a;
class Test<T>
{
    T obj;
    Test(T obj)
    {
        this.obj = obj;
    }
    public T getObject()
    {
        return this.obj;
    }
}

public class Generics
{
    public static void main (String[] args)
    {
        Test<Integer>iObj =new Test<Integer>(3);
        System.out.println(iObj.getObject());
        Test<String>sObj=new Test<String>("This is my Lucky Number");
        System.out.println(sObj.getObject());
    }
}
```

**Output:**

```
3  
This is my Lucky Number
```

**B. Write a Java Program to demonstrate Generic Methods.****Code:**

```
package a;  
public class GenericsMethod  
{  
    public static < E >void printArray(E[] elements)  
    {  
        for ( E element : elements)  
        {  
            System.out.println(element );  
        }  
        System.out.println();  
    }  
    public static void main( String args[] )  
    {  
        Integer[]intArray={ 10,20,30,40,50};  
        Character[] charArray = { 'A', 'D', 'V', 'A', 'N','C','E','D','I','T' };  
        System.out.println( "Printing Integer Array" );  
        printArray( intArray );  
        System.out.println( "Printing Character Array" );  
        printArray( charArray);  
    }  
}
```

**Output:**

```
Printing Integer Array  
10  
20  
30  
40  
50  
  
Printing Character Array  
A  
D  
V  
A  
N  
C  
E  
D  
I  
T
```

**C. Write a Java Program to demonstrate Wildcards in Java Generics.****Code:**

```
package a;  
class Test<T>
```

```

{
    T obj;
    Test(T obj)
    {
        this.obj = obj;
    }
    public T getObject()
    {
        return this.obj;
    }
}

public class Generics
{
    public static void main (String[] args)
    {
        Test<Integer>iObj =new Test<Integer>(3);
        System.out.println(iObj.getObject());
        Test<String>sObj=new Test<String>("This is my Lucky Number");
        System.out.println(sObj.getObject());
    }
}

```

**Output:**

```

drawing rectangle
drawing circle
drawing circle

```



## Practical No 2

### Aim: Assignments on List Interface

#### Description:

List in Java provides the facility to maintain the *ordered collection*. It contains the index-based methods to insert, update, delete and search the elements. It can have the duplicate elements also. We can also store the null elements in the list.

List interface is the child interface of Collection interface. It inhibits a list type data structure in which we can store the ordered collection of objects. It can have duplicate values.

List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

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

#### Code:

```
package b;
import java.util.*;
public class ListExample
{
    public static void main(String args[])
    {
        //Creating a list of elements
        ArrayList<String> list=new ArrayList<String>(); list.add("Java");
        list.add("Web Technology");
        list.add("Maths");
        //traversing the list of elements using for-each loop
        for(String s:list)
        {
            System.out.println(s);
        }
    }
}
```

#### Output:

```
Java
Web Technology
Maths
```

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

Code:

```
package b;
import java.util.*;
public class ListExample2{
public static void main(String args[]){
List<String> al=new ArrayList<String>();
    al.add("Amit");
    al.add("Vijay");
    al.add("Kumar");
    al.add(1,"Sachin");
    ListIterator<String> itr=al.listIterator();
    System.out.println("Traversing elements in forward direction");
    while(itr.hasNext()){

        System.out.println("index:"+itr.nextIndex()+" "+itr.next());
    }
    System.out.println("Traversing elements in backward direction");
    while(itr.hasPrevious()){

        System.out.println("index:"+itr.previousIndex()+" "+itr.previous());
    }
}
}
```

**Output:**

```
Traversing elements in forward direction
index:0 Amit
index:1 Sachin
index:2 Vijay
index:3 Kumar

Traversing elements in backward direction
index:3 Kumar
index:2 Vijay
index:1 Sachin
index:0 Amit
```

## **Practical No 3**

### **AIM : Assignments on Set Interface**

#### **Description:**

A Set is a Collection that cannot contain duplicate elements. It models the mathematical set abstraction.

The Set interface contains only methods inherited from Collection and adds the restriction that duplicate elements are prohibited.

Set also adds a stronger contract on the behavior of the equals and hashCode operations, allowing Set instances to be compared meaningfully even if their implementation types differ.

Set Interface in Java is present in java.util package. It extends the Collection interface. It represents the unordered set of elements. We can store at most one null value in Set. Set is implemented by HashSet, LinkedHashSet, and TreeSet.

**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:

```
package c;
import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.Collections;
public class SetExample {

    public static void main(String[] args)
    {

        // creating LinkedHashSet object of type String
        LinkedHashSet<String> lhsCompanies = new LinkedHashSet<String>();

        // adding elements to LinkedHashSet object
        lhsCompanies.add("LinkedIn");
        lhsCompanies.add("Amazon");
        lhsCompanies.add("Google");
        lhsCompanies.add("Apple");
        lhsCompanies.add("Facebook");
        lhsCompanies.add("Oracle");
        lhsCompanies.add("Microsoft");

        // Iterating using enhanced for-loop
```

```

System.out.println("Insertion Order: Iterating LinkedHashSet\n");
for(String company : lhsCompanies)
{
System.out.println(company);
}

// convert to ArrayList
ArrayList<String> alCompanies = new ArrayList<String>(lhsCompanies);

// to reverse LinkedHashSet contents
Collections.reverse(alCompanies);

// reverse order of LinkedHashSet contents
System.out.println("\n\nReverse Order of LinkedHashSet\n"); for(String company :
alCompanies) {
System.out.println(company);
}
}
}

```

### Output:

```

Insertion Order: Iterating LinkedHashSet
LinkedIn
Amazon
Google
Apple
Facebook
Oracle
Microsoft

Reverse Order of LinkedHashSet
Microsoft
Oracle
Facebook
Apple
Google
Amazon
LinkedIn

```

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

- **Add items in the set.**
- **Insert items of one set in to other set.**
- **Remove items from the set**
- **Search the specified item in the set**

### Code:

```
package c;
import java.util.HashSet;
import java.util.Iterator;

public class SetExampleb
{
    public static void main(String[] args) {
        HashSet<String> hset = new HashSet<String>();
        hset.add("Apple");
        hset.add("Mango");
        hset.add("Strawberry");
        System.out.println("Set before addAll:" + hset);

        hset.add("Welcome");
        System.out.println("Set after adding 1 elemrnt " + hset);

        HashSet<String> hset2 = new HashSet<String>(); hset2.add("Black-tea");
        hset2.add("Black-coffee");

        hset.addAll(hset2);
        System.out.println("Set after add_All: " + hset);

        hset.remove("Welcome");

        System.out.println("Set after removing elements: "
            + hset);

        System.out.println("Does the Set contains 'Item2'? "
            + hset.contains("Item2"));

        System.out.println("Does the Set contains ' Black-tea '? "
            + hset.contains("Black-tea"));
    }
}
```

### **Output:**

---

```
Set before addAll:[Apple, Strawberry, Mango]
Set after adding 1 elemrnt [Apple, Strawberry, Mango, Welcome]
Set after add_All: [Apple, Strawberry, Mango, Welcome, Black-tea, Black-coffee]
Set after removing elements: [Apple, Strawberry, Mango, Black-tea, Black-coffee]
Does the Set contains 'Item2'? false
Does the Set contains 'Black-tea'? true
```

## Practical No 4

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

- **Add items in the map.**
- **Remove items from the map**
- **Search specific key from the map**
- **Get value of the specified key**
- **Insert map elements of one map in to other map.**
- **Print all keys and values of the map.**

### Description:

The Map interface present in java.util package represents a mapping between a key and a value. The Map interface is not a subtype of the Collection interface. Therefore it behaves a bit differently from the rest of the collection types.

A map contains values on the basis of key, i.e. key and value pair. Each key and value pair is known as an entry. A Map contains unique keys.

A Map is useful if you have to search, update or delete elements on the basis of a key.

### Code:

```
package d;
import java.util.*;
import com.sun.tools.classfile.CharacterRangeTable_attribute.Entry;

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

        // Creating an empty HashMap
        HashMap<Integer, String> hash_map = new HashMap<Integer, String>();

        // Mapping string values to int keys
        hash_map.put(10, "Apple"); hash_map.put(15, "Banana"); hash_map.put(20, "Berries");
        hash_map.put(25, "Melons"); hash_map.put(30, "Oranges");

        // Displaying the HashMap
        System.out.println("Initial Mappings are: " + hash_map);

        // Creating a new hash map and copying
        HashMap<Integer, String> hmap2 = new HashMap<Integer, String>();
```

```

// Add items in the map
hmap2.put(11, "Hello");
hmap2.put(22, "Hi");

// Copying one HashMap "hash_map" to another HashMap "hmap2"
hmap2.putAll(hash_map);

// Displaying the final HashMap
System.out.println("The new map looks like this: " + hmap2);

//key-based removal hmap2.remove(30);
System.out.println("Updated list of elements: "+hmap2);

// Checking for the key_element '20'
System.out.println("Is the key '20' present? " + hash_map.containsKey(20));

// Checking for the key_element '5'
System.out.println("Is the key '5' present? " + hash_map.containsKey(5));
// Getting values from HashMap
String val=hmap2.get(11);
System.out.println("Get value of the specified key(11): "+ val);
}
}

```

## Output:

---

```

Initial Mappings are: {20=Berries, 25=Melons, 10=Apple, 30=Oranges, 15=Banana}
The new map looks like this: {20=Berries, 22=Hi, 25=Melons, 10=Apple, 11=Hello, 30=Oranges, 15=Banana}
Updated list of elements: {20=Berries, 22=Hi, 25=Melons, 10=Apple, 11=Hello, 30=Oranges, 15=Banana}
Is the key '20' present? true
Is the key '5' present? false
Get value of the specified key(11): Hello

```

## **Practical No 5**

### **Aim: Assignments on Lambda Expression**

#### **Description:**

Lambda expression is a new and important feature of Java which was included in Java SE 8. It provides a clear and concise way to represent one method interface using an expression. It is very useful in collection library. It helps to iterate, filter and extract data from collection.

The Lambda expression is used to provide the implementation of an interface which has functional interface. It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation. Here, we just write the implementation code. Java lambda expression is treated as a function, so compiler does not create .class file.

#### **A. Write a Java program using Lambda Expression to print "Hello World".**

##### Code:

```
package e;
@FunctionalInterface
interface MyFunctionalInterface
{
//A method with no parameter
public String sayHello();
}

public class Lamda1
{
public static void main(String args[])
{
// lambda expression
    MyFunctionalInterface msg = () ->
    {
return "Hello World";
};
System.out.println(msg.sayHello());
}
}
```

##### **Output:**

```
Hello World
```



**B. Write a Java program using Lambda Expression with single parameters.**

Code:

```
package e;
@FunctionalInterface
interface MyFun
{
//A method with single parameter
public int incrementByFive(int a);
}

public class Lamda2
{
public static void main(String args[])
{
// lambda expression with single parameter num
MyFun f = (num) -> num+5;
System.out.println("Lambda Expression With Single Parameter Numbers");
System.out.println("Addition of Number :"+ f.incrementByFive(70));
}
}
```

**Output:**

```
Lambda Expression With Single Parameter Numbers
Addition of Number :75
```

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

Code:

```
package e;
interface Addable
{ // Multiple parameters in lambda expression
int add(int a,int b);
}
public class Lamda3
{
public static void main(String[] args)
{
Addable ad1=(a,b)->(a+b); System.out.println("10+20=" +ad1.add(25,50));
// Multiple parameters with data type in lambda expression
Addable ad2=(int a,int b)->(a+b); System.out.println("100+200= "+ad2.add(120,50));
}
```

```
}  
}
```

**Output:**

```
10+20=75  
100+200= 170
```

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

- a. Convert Fahrenheit to Celcius**
- b. Convert Kilometers to Miles**

**Code:**

```
package e;  
@FunctionalInterface interface Converter  
{  
    double convert(double input);  
}  
  
public class Lamda4  
{  
    public static void main(String[] args)  
    {  
        // Convert Fahrenheit to Celsius  
        System.out.println("Fahrenheit to Celsius: "+convert(input -> (input-32)*5.0/9.0,  
98.6));  
  
        // Convert Kilometers to Miles  
        System.out.println("Kilometers to Miles: "+convert(input -> input/1.609344, 10));  
    }  
    static double convert(Converter converter, double input)  
    {  
        return converter.convert(input);  
    }  
}
```

**Output:**

```
Fahrenheit to Celsius: 37.0  
Kilometers to Miles: 6.2137119223733395
```

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

Code:

```
package e;
interface Sub
{
    int sub(int a,int b);
}

public class Lamda5
{
    public static void main(String[] args) {

        Sub ad=(a,b)->(a-b);
        System.out.println(ad.sub(40,15));

        // Lambda expression with return keyword.
        Sub ad2=(int a,int b)->
        {
            return (a-b);
        };
        System.out.println(ad2.sub(300,170));
    }
}
```

**Output:**

```
25
130
```

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

Code:

```
package e;
@FunctionalInterface
interface Sayable{
    String say(String message);
}

public class Lamda6
{
    public static void main(String[] args)
    {

        // You can pass multiple statements in lambda expression
        Sayable person = (message)-> {
```

```
String str1 = "Hi, i am yashita parab. ";  
  
String str2 = str1 + message;  
return str2;  
};  
System.out.println(person.say("currently perceiving MCA. "));  
}  
}
```

**Output:**

---

*Hi, i am yashita parab. currently perceiving MCA.*

## **Practical No 6**

### **Aim: Assignments based on web application development using JSP**

#### **Description:-**

**Java Server Pages :-** JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc. A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

#### **Advantages of JSP :-**

There are many advantages of JSP over the Servlet. They are as follows :-

- 1. Extension to Servlet :-** JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.
- 2. Easy to maintain :-** JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.
- 3. Fast Development:-** No need to recompile and redeploy :- If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application
- 4. Less code than Servlet :-** In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

Syntax available in JSP are following :-

- 1. Declaration Tag :-** It is used to declare variables. Syntax:- `<%! Dec var %>`
- 2. Java sScriptlet :-** It allows us to add any number of JAVA code, variables and expression Syntax:- `<% java code %>`
- 3. JSP Expression :-** It evaluates and convert the expression to a string. Syntax:- `<%= expression %>`
- 4. JAVA Comments :-** It contains the text that is added for information which has Syntax:- `<% -- JSP Comments %>`

**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:

```
<% @ page import="java.io.*,java.util.*,java.sql.*"%>
<% @ page import="javax.servlet.http.*,javax.servlet.*" %>
<% @ taglib uri="http://java.sun.com/jstl/core_rt" prefix="c"%>
<% @ taglib uri="http://java.sun.com/jstl/sql_rt" prefix="sql"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Telephone Directory</title>
</head>
<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/telephn" user="root" password="Vision@2022"/>
<sql:update dataSource="${snapshot}" var="result">
INSERT INTO TELEPHONE VALUES('Sandhya',9184567418);
</sql:update>
<sql:query dataSource="${snapshot}" var="result"> SELECT * from telephone;
</sql:query>
<table border="1" width="100%">
<tr>
<th>Person Name</th>
<th>Phone Number</th>
</tr>
<c:forEach var="row" items="${result.rows}">
<tr>
<td><c:out value="${row.name}" /></td>
<td><c:out value="${row.phone_no}" /></td>
</tr>
</c:forEach>
</table>
</html>
```

## Output:

Person Name	Phone Number
Sandhya	84567418
Sandhya	9184567418
Rahul	9778627977
Sagar	9150675824
Babita	8775025469
Sandhya	9184567418
Sandhya	9184567418
Sandhya	9184567418

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

Code:

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"% >
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>JSP Registration Form</title>
</head>
<body>
<center>
<h1 style="color:Crimson">JSP Registration Form</h1>
<form action="" method="post">
<table style="width: 50%">
<tr>
<td>First Name</td>
<td><input type="text" name="first_name"/></td>
</tr>
<tr>
<td>Last Name</td>
<td><input type="text" name="last_name"/></td>
</tr>
<tr>
<td>UserName</td>
<td><input type="text" name="username"/></td>
</tr>
<tr>
<td>Password</td>
```

```

<td><input type="password" name="password" /></td>
</tr>
<tr>
<td>Address</td>
<td><input type="text" name="address" /></td>
</tr>
<tr>
<td>Contact No</td>
<td><input type="text" name="contact" /></td>
</tr></table><br>
<input type="submit" value="Submit" /></form>
</center>
</body>
</html>

```

**Output:**

## JSP Registration Form

First Name	<input type="text" value="Rahul"/>
Last Name	<input type="text" value="Karwar"/>
UserName	<input type="text" value="rahulk19@gmail.com"/>
Password	<input type="password" value="••••••••••"/>
Address	<input type="text" value="Dombivli"/>
Contact No	<input type="text" value="9850120569"/> ×



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

Code:

**Form.html**

```
<!DOCTYPE html>
<html>
<body>
<form method="post" action="Insert.jsp"> Roll No:<br>
<input type="text" name="rollno">
<br> Name:<br>
<input type="text" name="name">
<br> Semester:<br>
<input type="text" name="semester">
<br> Course:<br>
<input type="text" name="course">
<br><br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

**Insert.jsp**

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<% @page import="java.sql.*,java.util.*"%>
<%
    String RollNo=request.getParameter("rollno");
    String Name=request.getParameter("name");
    String Semester=request.getParameter("semester");
    String Course=request.getParameter("course");
    try
    {
        Class.forName("com.mysql.jdbc.Driver"); Connection conn=
        DriverManager.getConnection("jdbc:mysql://localhost:3306/StudentMaster",
"root", "Vision@2022");
        Statement st=conn.createStatement();
        int i=st.executeUpdate("insert into
students(RollNo,Name,Semester,Course)values('"+RollNo+"','"+Name+"','"+Semester+
"','"+Course+"')");
        out.println("Data is successfully inserted!");
    }
    catch(Exception e)
    {
```

```

        System.out.print(e); e.printStackTrace();
    }
%>

```

## Output:

Roll No:

Name:

Semester:

Course:

Data is successfully inserted!

id	RollNo	Name	Semester	Course
2	55	Rahul Karwar	1st	JAVA

## DisplayStudent.jsp

```

<% @page import="java.sql.DriverManager"%>
<% @page import="java.sql.ResultSet"%>
<% @page import="java.sql.Statement"%>
<% @page import="java.sql.Connection"%>
<%
    String driver = "com.mysql.jdbc.Driver";
    String connectionUrl = "jdbc:mysql://localhost:3306/";
    String database = "StudentMaster";
    String userid = "root";
    String password= "Vision@2022";
    try
    {
        Class.forName(driver);
    }
    catch (ClassNotFoundException e)
    {
        e.printStackTrace();
    }
    Connection connection = null;

```

```

        Statement statement = null;
        ResultSet resultSet = null;
    %>
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<body>
<h1>Retrieve data from database in jsp</h1>
<table border="1">
<tr>
    <td>Roll No</td>
    <td>Name</td>
    <td>Semester</td>
    <td>Course</td>
    <td>Operations</td>
</tr>
<%
    try
    {
        connection = DriverManager.getConnection(connectionUrl+database, userid,
password);
        statement=connection.createStatement();
        String sql ="select * from students";
        resultSet = statement.executeQuery(sql);
        int i=0;
        while(resultSet.next())
        {
            %>
            <tr>
                <td><%=resultSet.getString("Rollno") %></td>
                <td><%=resultSet.getString("Name") %></td>
                <td><%=resultSet.getString("Semester") %></td>
                <td><%=resultSet.getString("Course") %></td>
                <td><a
href="DeleteStudent.jsp?id=<%=resultSet.getString("id") %>"><button type="button"
class="delete">Delete</button></a></td>
            </tr>
            <%
                i++;
            }
            connection.close();
        }
    catch (Exception e)
    {

```

```

        e.printStackTrace();
    }
%>
</table>
</body>
</html>

```

### **DeleteStudent.jsp**

```

<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<% @page import="java.sql.*,java.util.*"%>
<%
    String id=request.getParameter("id");
    try
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/StudentMaster", "root",
"Vision@2022");
        Statement st=conn.createStatement();
        int i=st.executeUpdate("DELETE FROM students WHERE id="+id);
        out.println("Data Deleted Successfully!");
    }
    catch(Exception e)
    {
        System.out.print(e); e.printStackTrace();
    }
%>

```

### **Output:**

## **Retrieve data from database in jsp**

Roll No	Name	Semester	Course	Operations
55	Rahul Karwar	1st	JAVA	Delete
12	Babita Biswas	1st	SPM	Delete
27	Sachin Dubey	1st	Advanced DBMS	Delete
116	Saurabh Shelar	1st		Delete
116	Saurabh Shelar	1st	Web Technology	Delete
21	Paresh Dhande	1st	Maths	Delete

Data Deleted Successfully!

id	RollNo	Name	Semester	Course
2	55	Rahul Karwar	1st	JAVA
3	12	Babita Biswas	1st	SPM
4	27	Sachin Dubey	1st	Advanced DBMS
7	116	Saurabh Shelar	1st	Web Technology
8	21	Paresh Dhande	1st	Maths

**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:**

1. 1 to 7 year at 5.35%
2. 8 to 15 year at 5.5%
3. 16 to 30 year at 5.75%

Code:

#### **LoanCalcForm.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>Loan Calculator</title>
</head>
<body>
<form action="LoanCalc.jsp">
<p>Enter Period of Time in years :- <input type="text" name="time" required></p>
<p>Enter Principal Loan Amount :- <input type="text" name="amt" required></p>
<p><input type="submit" value="Submit"></p>
</form>
</body>
</html>
```

#### **LoanCalc.jsp**

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

```

<title>Loan Calculator</title>
</head>
<body>
<%!
    String StrFormat(double a)
    {
        String res = String.format("%.2f", a);
        return res;
    }
%>
<%
    double P = Double.parseDouble(request.getParameter("amt"));
    double Year = Double.parseDouble(request.getParameter("time"));
    double N= Year *12;
    double tamt=0, intrest=0, emi=0, R=0;
    if( Year > 15 && Year < 31)
    {
        R=0.0575/12;
    }
    else if( Year > 7 && Year < 16)
    {
        R=0.055/(12);
    }
    else if( Year > 0 && Year < 8)
    {
        R=0.0535/12;
    }
    emi = (P * R * Math.pow((1+R),N))/(Math.pow(1+R, N)-1);
    out.print("Monthly Payment (EMI) :-"+StrFormat(emi)+"<br>");
    out.print("Total Payment :-" + StrFormat((emi*12)*Year)+"<br>");
    double balance = P;
    double principal, interest;
    out.print("<br><table border="+1+">");
    out.print("<tr><td> Payment# </td><td> Interest Paid </td><td> Principal Paid
</td><td>   Loan Balance </td></tr>");
    for (int i = 1; i <= N; i++)
    {
        interest = R * balance;
        principal = emi - interest;
        balance = balance -principal;
        out.print("<tr><td>" + i +"</td><td>" + StrFormat(interest)
+"</td><td>" +StrFormat(principal)+"</td><td>" + StrFormat(balance)+"</td></tr>");
    }
    out.print("</table>");

```

```
%>  
</body>  
</html>
```

### Output:

Enter Period of Time in years :-

Enter Principal Loan Amount :-

Monthly Payment (EMI) :-8805.66

Total Payment :-211335.86

Payment#	Interest Paid	Principal Paid	Loan Balance
1	891.67	7913.99	192086.01
2	856.38	7949.28	184136.73
3	820.94	7984.72	176152.01
4	785.34	8020.32	168131.69
5	749.59	8056.07	160075.62
6	713.67	8091.99	151983.63
7	677.59	8128.07	143855.56
8	641.36	8164.30	135691.26
9	604.96	8200.70	127490.55
10	568.40	8237.27	119253.29
11	531.67	8273.99	110979.30
12	494.78	8310.88	102668.42
13	457.73	8347.93	94320.49
14	420.51	8385.15	85935.34
15	383.13	8422.53	77512.81
16	345.58	8460.08	69052.73
17	307.86	8497.80	60554.93
18	269.97	8535.69	52019.24
19	231.92	8573.74	43445.50
20	193.69	8611.97	34833.53
21	155.30	8650.36	26183.17
22	116.73	8688.93	17494.24
23	78.00	8727.67	8766.58
24	39.08	8766.58	-0.00



**E. 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**

Code:

**StudentFormSC.jsp**

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPEhtml>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="StudentSC.jsp">
<table>
    <tr>
        <td colspan="2" align="center">Application for Change of Study
Center</td>
    </tr>
    <tr>
        <td>Enter Student Name:- </td>
        <td><input type="text" name="StdName" ></td>
    </tr>
    <tr>
        <td>Enter Student Application Id:-</td>
        <td><input type="text" name="StdId"></td>
    </tr>
    <tr>
        <td>Enter Branch Name:-</td>
        <td><input type="text" name="Branch"></td>
    </tr>
    <tr>
        <td colspan="2">Select Current Study Center:-</td>
    <td>
        <input type="radio" name="CurrentCenter" value="Mumbai"
required>Mumbai<br>
        <input type="radio" name="CurrentCenter" value="Pune"
required>Pune<br>
        <input type="radio" name="CurrentCenter" value="Delhi"
required>Delhi<br>
        <input type="radio" name="CurrentCenter" value="Kolkata"
required>Kolkata<br>
```

```

                <input type="radio" name="CurrentCenter" value="Chennai"
required>Chennai<br>
                <input type="radio" name="CurrentCenter" value="Hydrabad"
required>Hydrabad<br>
            </td>
        </tr>
        <tr>
            <td valign="top" >Select New Study Center:-</td>
            <td>
                <input type="radio" name="NewCenter" value="Mumbai"
required>Mumbai<br>
                <input type="radio" name="NewCenter" value="Pune"
required>Pune<br>
                <input type="radio" name="NewCenter" value="Delhi"
required>Delhi<br>
                <input type="radio" name="NewCenter" value="Kolkata"
required>Kolkata<br>
                <input type="radio" name="NewCenter" value="Chennai"
required>Chennai<br>
                <input type="radio" name="NewCenter" value="Hydrabad"
required>Hydrabad<br>
            </td>
        </tr>
        <tr>
            <td>Enter Reason for Change of Study Center:- </td>
            <td><input type="text" name="Reason"></td>
        </tr>
        <tr>
            <td colspan="2" align="center" ><input type="submit"
value="Submit"></td>
        </tr>
    </table>
</form>
</body>
</html>

```

### **StudentSC.jsp**

```

<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<% @ page import="java.util.Date" %>
<!DOCTYPEhtml>
<html>
<head>
<meta charset="ISO-8859-1">

```

```

<title>Insert title here</title>
</head>
<body>
<%
    Date date= new Date();
    String Date= date.getDate()+"/"+(date.getMonth()+1)+"/"+date.getYear();
    String Name= request.getParameter("StdName");
    String Id= request.getParameter("StdId");
    String Branch= request.getParameter("Branch");
    String CurrentCenter= request.getParameter("CurrentCenter");
    String NewCenter= request.getParameter("NewCenter");
    String Reason= request.getParameter("Reason");
%>
<table>
    <tr>
        <td><%= Date%></td>
    </tr>
    <tr>
        <td>The Principle,</td>
    </tr>
    <tr>
        <td>University Of Mumbai</td>
    </tr>
    <tr>
        <td>VidyaNagari, KalaGhoda,</td>
    </tr>
    <tr>
        <td>Fort, Mumbai</td>
    </tr>
    <tr>
        <td><u>Maharashtra</u> 400032</td>
    </tr>
    <tr>
        <td>&emsp;&emsp;Sub: Request for Change Examination Center</td>
    </tr>
    <tr>
        <td>Respected Sir,</td>
    </tr>
    <tr>
        <td>&emsp;&emsp;I am <%= Name%> writing this to you so that I can
share my problem, Sir you have allotted me</td></tr>
    <tr>
        <td>&emsp;&emsp;<%= CurrentCenter %> as study center and the center
allotted to me is very far from my house and as I have a serious</td>

```

```

        </tr>
        <tr>
            <td>&emsp;&emsp;transportation issue it will not be possible for me to go
there due to <%= Reason %> So I want you to change</td>
        </tr>
        <tr>
            <td>&emsp;&emsp;my examination center from <%= CurrentCenter %>
center to <%= NewCenter %> center</td>
        </tr>
        <tr>
            <td>&emsp;&emsp;It would be so much helpful for me.</td>
        </tr>
        <tr>
            <td>Thanks,</td>
        </tr>
        <tr>
            <td>Sincerely,</td>
        </tr>
        <table border="1">
            <tr>
                <td>Name of Student:-</td>
                <td><%= Name%></td>
            </tr>
            <tr>
                <td>Application ID of Student :-</td>
                <td><%= Id%></td>
            </tr>
            <tr>
                <td>Branch of Student :-</td>
                <td><%= Branch%></td>
            </tr>
        </table>
    </table>
</body>
</html>

```

## Output:

Application for Change of Study Center	
Enter Student Name:-	<input type="text" value="Rahul Tanaji Karwar"/>
Enter Student Application Id:-	<input type="text" value="22055"/>
Enter Branch Name:-	<input type="text" value="Thane"/>
Select Current Study Center:-	<input checked="" type="radio"/> Mumbai <input type="radio"/> Pune <input type="radio"/> Delhi <input type="radio"/> Kolkata <input type="radio"/> Chennai <input type="radio"/> Hyderabad
Select New Study Center:-	<input type="radio"/> Mumbai <input checked="" type="radio"/> Pune <input type="radio"/> Delhi <input type="radio"/> Kolkata <input type="radio"/> Chennai <input type="radio"/> Hyderabad
Enter Reason for Change of Study Center:-	<input type="text" value="Shifting"/>
<input type="button" value="Submit"/>	

17/3/122

The Principle,  
University Of Mumbai  
VidyaNagari, KalaGhoda,  
Fort, Mumbai  
Maharashtra 400032

Sub: Request for Change Examination Center

Respected Sir,

I am Rahul Tanaji Karwar writing this to you so that I can share my problem, Sir you have allotted me Mumbai as study center and the center allotted to me is very far from my house and as I have a serious transportation issue it will not be possible for me to go there due to Shifting So I want you to change my examination center from Mumbai center to Pune center

It would be so much helpful for me.

Thanks,

Sincerely,

Name of Student:-	Rahul Tanaji Karwar
Application ID of Student :-	22055
Branch of Student :-	Thane

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

Code:

**Form.html**

```
<!DOCTYPE html>
<html>
<body>
<form method="post" action="Insert.jsp"> Roll No:<br>
<input type="text" name="rollno">
<br> Name:<br>
<input type="text" name="name">
<br> Semester:<br>
<input type="text" name="semester">
<br> Course:<br>
<input type="text" name="course">
<br><br>
<input type="submit" value="submit">
</form>
</body>
</html>
```

**Insert.jsp**

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<% @page import="java.sql.*,java.util.*"%>
<%
    String RollNo=request.getParameter("rollno");
    String Name=request.getParameter("name");
    String Semester=request.getParameter("semester");
    String Course=request.getParameter("course");
    try
    {
        Class.forName("com.mysql.jdbc.Driver"); Connection conn=
        DriverManager.getConnection("jdbc:mysql://localhost:3306/StudentMaster",
"root", "Vision@2022");
        Statement st=conn.createStatement();
        int i=st.executeUpdate("insert into
students(RollNo,Name,Semester,Course)values('"+RollNo+"','"+Name+"','"+Semester+
"','"+Course+"')");
        out.println("Data is successfully inserted!");
    }
    catch(Exception e)
    {
```

```

        System.out.print(e); e.printStackTrace();
    }
%>

```

### Output:

Roll No:

Name:

Semester:

Course:



Data is successfully inserted!

id	RollNo	Name	Semester	Course
2	55	Rahul Karwar	1st	JAVA

### DisplayStudent.jsp

```

<% @page import="java.sql.DriverManager"%>
<% @page import="java.sql.ResultSet"%>
<% @page import="java.sql.Statement"%>
<% @page import="java.sql.Connection"%>
<%
    String driver = "com.mysql.jdbc.Driver";
    String connectionUrl = "jdbc:mysql://localhost:3306/";
    String database = "StudentMaster";
    String userid = "root";
    String password= "Vision@2022";
    try
    {
        Class.forName(driver);
    }
    catch (ClassNotFoundException e)
    {
        e.printStackTrace();
    }
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;

```

```

%>
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
<body>
<h1>Retrieve data from database in jsp</h1>
<table border="1">
<tr>
<td>Roll No</td>
<td>Name</td>
<td>Semester</td>
<td>Course</td>
<td>Operations</td>
</tr>
<%
    try
    {
        connection = DriverManager.getConnection(connectionUrl+database, userid,
password);
        statement=connection.createStatement();
        String sql ="select * from students";
        resultSet = statement.executeQuery(sql);
        int i=0;
        while(resultSet.next())
        {
            %>
            <tr>
                <td><%=resultSet.getString("Rollno") %></td>
                <td><%=resultSet.getString("Name") %></td>
                <td><%=resultSet.getString("Semester") %></td>
                <td><%=resultSet.getString("Course") %></td>
                <td><a
href="DeleteStudent.jsp?id=<%=resultSet.getString("id") %>"><button type="button"
class="delete">Delete</button></a></td>
                </tr>
            <%
                i++;
            }
            connection.close();
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
%>

```



```
%>
</table>
</body>
</html>
```

### **DeleteStudent.jsp**

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<% @page import="java.sql.*,java.util.*"%>
<%
    String id=request.getParameter("id");
    try
    {
        Class.forName("com.mysql.jdbc.Driver");
        Connection
conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/StudentMaster", "root",
"Vision@2022");
        Statement st=conn.createStatement();
        int i=st.executeUpdate("DELETE FROM students WHERE id="+id);
        out.println("Data Deleted Successfully!");
    }
    catch(Exception e)
    {
        System.out.print(e); e.printStackTrace();
    }
%>
```

**Output:**

## **Retrieve data from database in jsp**

Roll No	Name	Semester	Course	Operations
55	Rahul Karwar	1st	JAVA	Delete
12	Babita Biswas	1st	SPM	Delete
27	Sachin Dubey	1st	Advanced DBMS	Delete
116	Saurabh Shelar	1st		Delete
116	Saurabh Shelar	1st	Web Technology	Delete
21	Paresh Dhande	1st	Maths	Delete

---

Data Deleted Successfully!

id	RollNo	Name	Semester	Course
2	55	Rahul Karwar	1st	JAVA
3	12	Babita Biswas	1st	SPM
4	27	Sachin Dubey	1st	Advanced DBMS
7	116	Saurabh Shelar	1st	Web Technology
8	21	Paresh Dhande	1st	Maths

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

Code:-

### Main.jsp

```
<% @ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title> JSP Example</title>
</head>
<body>
<% @ include file="header.html" %>
<% out.println("This is JSP Example"); %>
<% out.println("The number is "); %>
<% ! int num12 = 15; int num32 = 15; %>
<%= num12*num32 %>
Today's date: <%= (new java.util.Date()).toLocaleString()%>
<% @ include file="footer.html" %>
</body>
</html>
```

### header.html

```
<h1>This is Footer</h1>
```

### footer.html

```
<h1>This is Footer</h1>
```

**Output:**

# This is Header

This is JSP Example The number is 225 Today's date: Mar 17, 2022, 9:56:23 PM

# This is Footer

## **Practical No 7**

### **Aim: Assignment based on Spring Framework**

#### **Description:**

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

It is a lightweight, loosely coupled and integrated framework. Spring Framework is built on top of two design concepts ? Dependency Injection and Aspect Oriented Programming.

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

Code:

##### **HelloWorld.java**

```
package com.first;
```

```
public class HelloWorld
{
    public String message;
    public void setMessage(String message){
        this.message = message;
    }
    public void getMessage(){ System.out.println("Spring Framework Message: " + message);
    }
}
```

##### **MainApp.java**

```
package com.first;
```

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

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

```
public class MainApp
{
    public static void main(String[] args)
    {
        ApplicationContext context = new
        ClassPathXmlApplicationContext("com/first/Beans.xml");
```

```

HelloWorld obj = (HelloWorld) context.getBean("helloWorld"); obj.getMessage();
}
}

```

### **Beans.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-3.0.xsd">

<bean id = "helloWorld" class = "com.first.HelloWorld">
<property name = "message" value = "Hello World!"/>
</bean>
</beans>

```

### **Output:**

```

19:21:16.188 [main] DEBUG org.springfram
19:21:16.417 [main] DEBUG org.springfram
19:21:16.456 [main] DEBUG org.springfram

Spring Framework Message: Hello World!

```

## **B. Write a program to demonstrate dependency injection via setter method.**

### **Code:**

#### **Employee.java**

```

package com.two;

public class Employee
{
public int id; public String name; public String city;

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 String getCity() {
return city;
}
public void setCity(String city) {
this.city = city;
}
void display(){
System.out.println("\n"+id+" "+name+" "+city);
}
}

```

### **Test.java**

```

package com.two;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.*;

public class Test
{
public static void main(String[] args)
{
Resource r=new ClassPathResource("com/two/applicationContext.xml"); BeanFactory
factory=new XmlBeanFactory(r);
Employee e=(Employee)factory.getBean("obj"); e.display();
}
}

```

### **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: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="obj" class="com.two.Employee">
<property name="id">
<value>20</value>
</property>
<property name="name">
<value>Rahul</value>

```

```

</property>
<property name="city">
<value>Dombivli</value>
</property>
</bean>
</beans>

```

### **Output:**

```

19:31:31.032 [main] DEBUG c
19:31:31.038 [main] DEBUG c

20 Rahul Dombivli

```

### **C. Write a program to demonstrate dependency injection via Constructor.**

Code:

#### **Employee.java**

```

package com.three;

public class Employee
{
    private int id;
    private String name;
    public Employee() {System.out.println("def cons");
    }
    public Employee(int id)
    {
        this.id = id;
    }
    public Employee(String name)
    {
        this.name = name;
    }
    public Employee(int id, String name)
    {
        this.id = id;
        this.name = name;
    }
    void show()
    {
        System.out.println("\nEmployee ID:- "+id+"\nEmployee Name:- "+name);
    }
}

```

### **Test.java**

```
package com.three;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.*;

public class Test
{
    public static void main(String[] args)
    {
        Resource r=new ClassPathResource("com/three/application.xml"); BeanFactory
        factory=new XmlBeanFactory(r);
        Employee s=(Employee)factory.getBean("e"); s.show();
    }
}
```

### **application.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="e" class="com.three.Employee">
<constructor-arg value="110" type="int"></constructor-arg>
<constructor-arg value="Rahul"></constructor-arg>
</bean>
</beans>
```

### **Output:**

```
20:09:49.783 [main] DEBU
20:09:49.789 [main] DEBU
```

```
Employee ID:- 110
Employee Name:- Rahul
```

## **Practical No 8**

### **Aim: Assignment based Aspect Oriented Programming**

#### **Description:**

An aspect is a common feature that's typically scattered across methods, classes, object hierarchies, or even entire object models. It is behavior that looks and smells like it should have structure, but you can't find a way to express this structure in code with traditional object-oriented techniques.

**Aspect oriented programming(AOP)** as the name suggests uses aspects in programming. It can be defined as the breaking of code into different modules, also known as [modularisation](#), where the aspect is the key unit of modularity. Aspects enable the implementation of crosscutting concerns such as- transaction, logging not central to business logic without cluttering the code core to its functionality. It does so by adding additional behaviour that is the advice to the existing code. For example- Security is a crosscutting concern, in many methods in an application security rules can be applied, therefore repeating the code at every method, define the functionality in a common class and control were to apply that functionality in the whole application.

#### **A. Write a program to demonstrate Spring AOP – before advice.**

##### Code:

##### **App.java**

```
package com.ram;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.ram.EmployeeService;

public class App
{
    public static void main(String[] args)
    {
        ApplicationContext context = new
        ClassPathXmlApplicationContext("com/ram/applicationContext.xml");
        System.out.println(" "); EmployeeService employeeService =
        context.getBean("employeeServiceProxy",
        EmployeeService.class);
        employeeService.displayEmployeeInfo();
    }
}
```



### **EmployeeService.java**

```
package com.ram;

public class EmployeeService
{
    public void displayEmployeeInfo()
    {
        System.out.println("\nDisplay Employee Information");
    }
}
```

### **LoggingAdvice.java**

```
package com.ram;
import java.lang.reflect.Method;
import org.springframework.aop.MethodBeforeAdvice;

public class LoggingAdvice implements MethodBeforeAdvice
{
    public void before(Method method, Object[] args, Object target)
        throws Throwable
    {
        System.out.println("Logging advice is applied before the method "
            + method.getName() + " in the target Object "
            + target.getClass().getName());
    }
}
```

### **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:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-4.3.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <bean id="employeeService" class="com.ram.EmployeeService"></bean>
    <bean id="loggingAdvice" class="com.ram.LoggingAdvice"></bean>

    <bean id="employeeServiceProxy"
        class="org.springframework.aop.framework.ProxyFactoryBean">
```

```
<property name="target" ref="employeeService"></property>
<property name="interceptorNames">
<list>
<value>loggingAdvice</value>
</list>
</property>
</bean>
</beans>
```

## Output:

```
Logging advice is applied before the method displayEmployeeInfo in the target Object com.ram.EmployeeService
Display Employee Information
```

## **B. Write a program to demonstrate Spring AOP – after advice.**

Code:

### **Logging.java**

```
package com.two;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.annotation.After;

@Aspect

public class Logging
{
    /** Following is the definition for a PointCut to select
     *    all the methods available. So advice will be called
     *    for all the methods.
     */
    @Pointcut("execution(* com.two.Student.getAge(..))")
    private void selectGetAge() { }

    /**
     *    This is the method which I would like to execute
     *    after a selected method execution.
     */
    @After("selectGetAge()") public void afterAdvice() {
        System.out.println("\nStudent profile setup completed.");
    }
}
```

### **Student.java**

```
package com.two;

public class Student
{
    private Integer age;
    private String name;

    public void setAge(Integer age) {
        this.age = age;
    }
    public Integer getAge() { System.out.println("Age : " + age ); return age;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getName() { System.out.println("Name : " + name ); return name;
    }
}
```

```

}
public void printThrowException(){ System.out.println("Exception raised");
}
}

```

### **MainApp.java**

```

package com.two;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

```

```

public class MainApp
{
public static void main(String[] args) {
    ApplicationContext context = new
    ClassPathXmlApplicationContext("com/two/Beans.xml");

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

    student.getName(); student.getAge();
}
}

```

### **Beans.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-3.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">

<aop:aspectj-autoproxy/>
<!-- Definition for student bean -->
<bean id = "student" class = "com.two.Student">
<property name = "name" value = "Rahul" />
<property name = "age" value = "22"/>
</bean>

<!-- Definition for logging aspect -->
<bean id = "logging" class = "com.two.Logging" />
</beans>

```

## Output:

```
Name : Rahul
Age : 22

Student profile setup completed.
```

## C. Write a program to demonstrate Spring AOP – around advice.

Code:

### A.java

```
package three;
public class A
{
    public void m()
    {
        System.out.println("actual business logic");
    }
}
```

### three.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="obj" class="three.A"></bean>
    <bean id="ba" class="three.AroundAdvisor"></bean>
    <bean id="proxy" class="org.springframework.aop.framework.ProxyFactoryBean">
        <property name="target" ref="obj"></property>
        <property name="interceptorNames">

            <list>
                <value>ba</value>
            </list>
        </property>
    </bean>
</beans>
```

### Test.java

**package** three;

```
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;
```

**public class** Test

```
{
public static void main(String[] args)
{
    Resource r=new ClassPathResource("three.xml"); BeanFactory factory=new
XmlBeanFactory(r);
```

```
    A a=factory.getBean("proxy",A.class);
    System.out.println("proxy class name: "+a.getClass().getName());
    a.m();
}
}
```

### AroundAdvisor.java

**package** three;

```
import    org.aopalliance.intercept.MethodInterceptor;
import    org.aopalliance.intercept.MethodInvocation;
public class AroundAdvisor implements MethodInterceptor{
```

@Override

```
public Object invoke(MethodInvocation mi) throws Throwable {
    Object obj;
    System.out.println("Additional concern before actual logic");
    obj=mi.proceed();
    System.out.println("Additional concern after actual logic");
    return obj;
}
}
```

## Output:

```
14:38:29.701 [main] DEBUG org.springframework.bea
14:38:29.707 [main] DEBUG org.springframework.bea
14:38:29.725 [main] DEBUG org.springframework.bea
14:38:29.853 [main] DEBUG org.springframework.bea
14:38:29.855 [main] DEBUG org.springframework.aop
proxy class name: three.A$$EnhancerBySpringCGLIB$
Additional concern before actual logic
actual business logic
Additional concern after actual logic
```

## **D. Write a program to demonstrate Spring AOP – after returning advice.**

Code:

### **A.java**

```
package two;
public class A
{
    public void m()
    {
        System.out.println("actual business logic");
    }
}
```

### **AfterAdvisor.java**

```
package two;
import java.lang.reflect.Method;
import org.springframework.aop.AfterReturningAdvice;

public class AfterAdvisor implements AfterReturningAdvice
{
    @Override
    public void afterReturning(Object returnValue, Method method, Object[] args, Object
target) throws Throwable
    {
        System.out.println("additional concern after returning advice");
    }
}
```

### **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: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="obj" class="two.A"></bean>
<bean id="ba" class="two.AfterAdvisor"></bean>

<bean id="proxy" class="org.springframework.aop.framework.ProxyFactoryBean">
<property name="target" ref="obj"></property>
<property name="interceptorNames">
<list>
<value>ba</value>
</list>
</property>
</bean>
</beans>
```

### **Test.java**

```
package two;

import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory; import
org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

public class Test
{
public static void main(String[] args)
{
Resource r=new ClassPathResource("application.xml"); BeanFactory factory=new
XmlBeanFactory(r);

A a=factory.getBean("proxy",A.class);
System.out.println("proxy class name: "+a.getClass().getName()); a.m();
}
}
```



### Output:

```
14:22:43.732 [main] DEBUG org.springframework.beans.factory
14:22:43.739 [main] DEBUG org.springframework.beans.factory
14:22:43.759 [main] DEBUG org.springframework.beans.factory
14:22:43.878 [main] DEBUG org.springframework.beans.factory
14:22:43.880 [main] DEBUG org.springframework.aop.framework
proxy class name: two.A$$EnhancerBySpringCGLIB$$6c6233a0
actual business logic
additional concern after returning advice
```

### **E. Write a program to demonstrate Spring AOP – after throwing advice.**

Code:

#### **Logging.java**

```
package com.five;
```

```
import org.aspectj.lang.annotation.*; @Aspect
```

```
public class Logging
```

```
{
```

```
public void beforeAdvice()
```

```
{
```

```
System.out.println("Going to setup student profile.");
```

```
}
```

```
public void afterAdvice(){ System.out.println("Student profile has been setup.");
```

```
}
```

```
public void afterReturningAdvice(Object retVal)
```

```
{
```

```
System.out.println("Returning:" + retVal.toString() );
```

```
}
```

```
@AfterThrowing(pointcut = "selectAll()", throwing = "ex")
```

```
public void AfterThrowingAdvice(IllegalArgumentException ex)
```

```
{
```

```
    System.out.println("There has been an exception: " + ex.toString());
```

```
}
```

```
}
```

### **Student.java**

**package** com.five;

**public class** Student { **private** Integer age; **private** String name;

**public void** setAge(Integer age) {

**this**.age = age;

}

**public** Integer getAge()

{

    System.*out*.println("Age : " + age );

**return** age;

}

**public void** setName(String name)

{

**this**.name = name;

}

**public** String getName()

{

    System.*out*.println("Name : " + name );

**return** name;

}

**public void** printThrowException()

{

    System.*out*.println("Exception raised");

**throw new** IllegalArgumentException();

}

}

### **MainApp.java**

**package** com.five;

**import** org.springframework.context.ApplicationContext;

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

**public class** MainApp {

**public static void** main(String[] args) {

ApplicationContext context = **new** ClassPathXmlApplicationContext("Bean.xml");

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

student.getName(); student.getAge(); student.printThrowException();

}

}

### **Bean.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-3.0.xsd
http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd" >
<aop:aspectj-autoproxy/>
<!-- Definition for student bean -->
<bean id = "student" class = "com.five.Student">

<property name = "name" value = "Zara" />
<property name = "age" value = "11"/>
</bean>
<!-- Definition for logging aspect -->
<bean id = "logging" class = "com.five.Logging"/>
</beans>
```

### **Output:**

```
15:49:05.702 [main] DEBUG org.springframework.context.support.ClassPathXr
15:49:05.989 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanD
15:49:06.021 [main] DEBUG org.springframework.beans.factory.support.Defai
15:49:06.136 [main] DEBUG org.springframework.beans.factory.support.Defai
15:49:06.195 [main] DEBUG org.springframework.aop.aspectj.annotation.Ref
15:49:06.197 [main] DEBUG org.springframework.aop.aspectj.annotation.Ref
15:49:06.197 [main] DEBUG org.springframework.aop.aspectj.annotation.Ref
15:49:06.211 [main] DEBUG org.springframework.aop.aspectj.annotation.Ref
Exception in thread "main" 15:49:06.363 [main] DEBUG org.springframework
Name : Zara
Age : 11
Exception raised
```

### **E. Write a program to demonstrate Spring AOP – pointcuts.**

Code:

#### **Logging.java**

**package** six;

```
import org.aspectj.lang.annotation.Aspect; import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.annotation.Before;
```

@Aspect

```
public class Logging {
```

```

/** Following is the definition for a PointCut to select
    * all the methods available. So advice will be called
    * for all the methods.

*/
//@PointCut("execution(* six.*(..))") @Pointcut("within(six.*)")

private void selectAll()
{}

/**
    * This is the method which I would like to execute
    * before a selected method execution.

*/ @Before("selectAll()")
public void beforeAdvice(){ System.out.println("Going to setup student profile.");
}
}

```

### **Student.java**

```

package six;

public class Student {
private Integer age;
private String name;

public void setAge(Integer age) {
this.age = age;
}

public Integer getAge() { System.out.println("Age : " + age ); return age;
}

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

public String getName() { System.out.println("Name : " + name ); return name;
}

public void printThrowException(){ System.out.println("Exception raised"); throw new
IllegalArgumentException();
}

```

```
}  
}
```

### MainApp.java

```
package six;
```

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

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

```
public class MainApp {
```

```
public static void main(String[] args)
```

```
{
```

```
ApplicationContext context =
```

```
new ClassPathXmlApplicationContext("Bean.xml");
```

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

```
student.getName(); student.getAge();
```

```
}
```

```
}
```

### Bean.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-3.0.xsd
```

```
http://www.springframework.org/schema/aop
```

```
http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">
```

```
<aop:aspectj-autoproxy/>
```

```
<!-- Definition for student bean -->
```

```
<bean id = "student" class = "six.Student">
```

```
<property name = "name" value = "Zara" />
```

```
<property name = "age" value = "11"/>
```

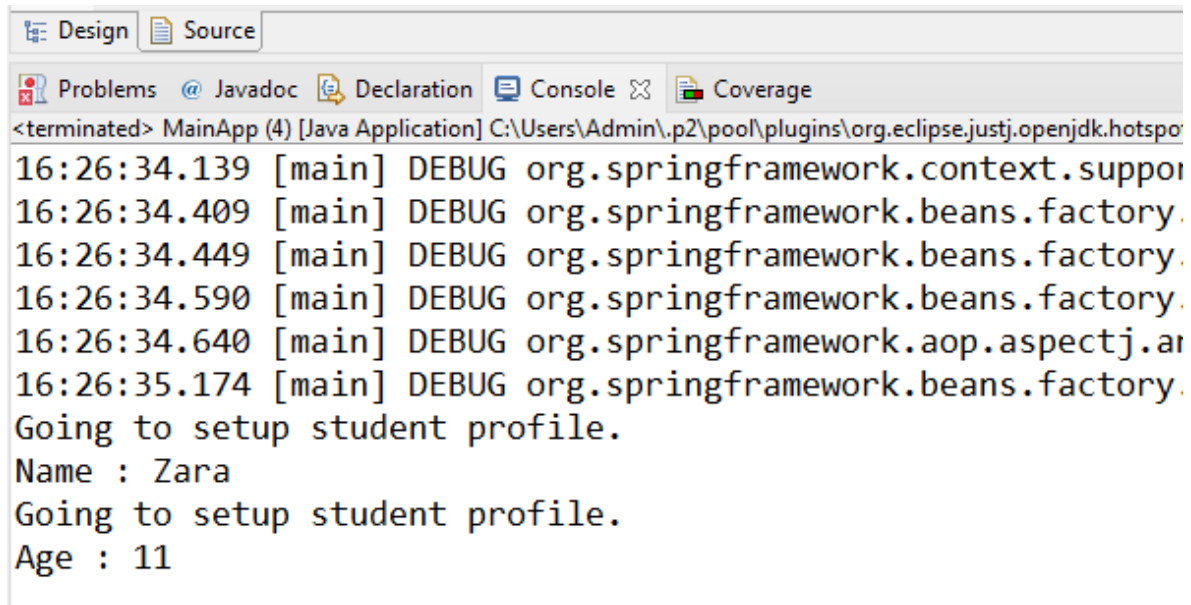
```
</bean>
```

```
<!-- Definition for logging aspect -->
```

```
<bean id = "logging" class = "six.Logging"/>
```

```
</beans>
```

## Output:

A screenshot of the Eclipse IDE's console window. The window has tabs for 'Design', 'Source', 'Problems', 'Javadoc', 'Declaration', 'Console', and 'Coverage'. The 'Console' tab is active, showing the output of a Java application. The output starts with a path and then shows several debug messages from the Spring Framework, followed by two lines of user input: 'Name : Zara' and 'Age : 11'.

```
<terminated> MainApp (4) [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot
16:26:34.139 [main] DEBUG org.springframework.context.support
16:26:34.409 [main] DEBUG org.springframework.beans.factory
16:26:34.449 [main] DEBUG org.springframework.beans.factory
16:26:34.590 [main] DEBUG org.springframework.beans.factory
16:26:34.640 [main] DEBUG org.springframework.aop.aspectj.ar
16:26:35.174 [main] DEBUG org.springframework.beans.factory
Going to setup student profile.
Name : Zara
Going to setup student profile.
Age : 11
```

## Practical No 9

### Aim: Assignment based Spring JDBC

#### Description:

While working with the database using plain old JDBC, it becomes cumbersome to write unnecessary code to handle exceptions, opening and closing database connections, etc. However, Spring JDBC Framework takes care of all the low-level details starting from opening the connection, prepare and execute the SQL statement, process exceptions, handle transactions and finally close the connection.

So what you have to do is just define the connection parameters and specify the SQL statement to be executed and do the required work for each iteration while fetching data from the database.

Spring JDBC provides several approaches and correspondingly different classes to interface with the database. I'm going to take classic and the most popular approach which makes use of **JdbcTemplate** class of the framework. This is the central framework class that manages all the database communication and exception handling.

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

Code:

##### MySpringJDBC.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-3.0.xsd ">
<!-- Initialization for data source -->
<bean id="dataSource"
class = "org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name = "driverClassName" value = "com.mysql.jdbc.Driver"/>
<property name = "url" value = "jdbc:mysql://localhost:3306/imcost?useSSL=false"/>
<property name = "username" value = "root"/>
<property name = "password" value = "1234"/>
</bean>
<!-- Definition for studentJdbcTemplate bean -->
<bean id = "studentJdbcTemplate"
class = "com.test.StudentJdbcTemplate">
<property name = "dataSource" ref = "dataSource" />
```

```
</bean>  
</beans>
```

### **StudentDAO.java**

```
package com.test;  
import java.util.List;  
import javax.sql.DataSource;  
  
public interface StudentDAO  
{  
public void setDataSource(DataSource ds);  
public void create(String name, Integer age);  
  
public Student getStudent(Integer id);  
  
public List<Student> listStudents();  
  
public void delete(Integer id);  
  
public void update(Integer id, Integer age);  
}  
}  
Student.java package com.test;  
public class Student {  
  
public static void main(String[] args) {  
public class Student {  
  
private Integer id;  
  
private String name; private Integer age; public Integer getId() { return id;  
}  
public void setId(Integer id) {  
this.id = id;  
}  
public String getName() {  
return name;  
}  
public void setName(String name) {  
this.name = name;  
}  
public Integer getAge() {  
return age;
```



```

}
public void setAge(Integer age) {
this.age = age;
}
}
}
}
}

```

### **Student.JdbcTemplate.java**

```

package com.test;
import java.util.List;
import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;

public class StudentJdbcTemplate implements StudentDAO
{
    private DataSource dataSource;
    private JdbcTemplate jdbcTemplateObject;

    @Override
    public void setDataSource(DataSource ds) {

        this.dataSource=ds;
        this.jdbcTemplateObject = new JdbcTemplate(dataSource);
    }

    @Override
    public void create(String name, Integer age) {
        String SQL = "insert into Student (name, age) values (?, ?)";
        jdbcTemplateObject.update( SQL, name, age);
        System.out.println("Created Record Name = " + name + " Age = " + age);
        return;
    }

    @Override
    public Student getStudent(Integer id) {

        return null;
    }

    @Override
    public List<Student> listStudents() {

        return null;
    }
}

```

```

@Override
public void delete(Integer id) {

String SQL = "delete from Student where id = ?";
jdbcTemplateObject.update(SQL, id);
System.out.println("Deleted Record with ID = " + id );
return;
}

@Override
public void update(Integer id, Integer age) {
String SQL = "update Student set age = ? where id = ?";
jdbcTemplateObject.update(SQL, age, id);
System.out.println("Updated Record with ID = " + id );
return;
}
}

```

### **MyMain.java**

```

package com.test;
import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

```

```

public class MyMain
{
public static void main(String[] args)
{
ApplicationContext context = new
ClassPathXmlApplicationContext("MySpringJDBC.xml"); StudentJDBCTemplate
studentJDBCTemplate
=(StudentJDBCTemplate)context.getBean("studentJDBCTemplate");

studentJDBCTemplate.create("King", 45);
studentJDBCTemplate.create("Queen", 42);
studentJDBCTemplate.create("Worker", 23); System.out.println("-----Records Creation "
);
studentJDBCTemplate.delete(2); System.out.println("-----Records Deleted " );
studentJDBCTemplate.update(3, 23); System.out.println("-----Records Updated   ");
}
}

```

## Output:

```
Problems @ Javadoc Declaration Console Coverage
<terminated> MyMain (1) [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_15.0.2.v20210201-0955\jre\bin\javaw.exe (Apr 7, 2021, 7:32:02
19:32:03.523 [main] DEBUG org.springframework.context.support.ClassPathXmlApplicationContext - Refreshi
19:32:03.810 [main] DEBUG org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loaded 2 bear
19:32:03.839 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creati
Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.D
19:32:03.935 [main] DEBUG org.springframework.jdbc.datasource.DriverManagerDataSource - Loaded JDBC dri
19:32:03.936 [main] DEBUG org.springframework.beans.factory.support.DefaultListableBeanFactory - Creati
19:32:03.958 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update
19:32:03.960 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL statement
19:32:03.963 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Fetching JDBC Connecti
19:32:03.963 [main] DEBUG org.springframework.jdbc.datasource.DriverManagerDataSource - Creating new JC
Created Record Name = King Age = 45
19:32:04.467 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update
19:32:04.467 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL statement
19:32:04.467 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Fetching JDBC Connecti
19:32:04.468 [main] DEBUG org.springframework.jdbc.datasource.DriverManagerDataSource - Creating new JC
Created Record Name = Queen Age = 42
19:32:04.559 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update
19:32:04.559 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL statement
19:32:04.560 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUtils - Fetching JDBC Connecti
19:32:04.560 [main] DEBUG org.springframework.jdbc.datasource.DriverManagerDataSource - Creating new JC
Created Record Name = Worker Age = 23
-----Records Creation-----
19:32:04.750 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Executing prepared SQL update
Created Record Name = Queen Age = 42
19:32:04.559 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Exec
19:32:04.559 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Exec
19:32:04.560 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUti
19:32:04.560 [main] DEBUG org.springframework.jdbc.datasource.DriverManager
Created Record Name = Worker Age = 23
-----Records Creation-----
19:32:04.750 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Exec
19:32:04.750 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Exec
19:32:04.750 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUti
19:32:04.750 [main] DEBUG org.springframework.jdbc.datasource.DriverManager
Deleted Record with ID = 2
-----Records Deleted-----
19:32:04.881 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Exec
19:32:04.881 [main] DEBUG org.springframework.jdbc.core.JdbcTemplate - Exec
19:32:04.881 [main] DEBUG org.springframework.jdbc.datasource.DataSourceUti
19:32:04.881 [main] DEBUG org.springframework.jdbc.datasource.DriverManager
Updated Record with ID = 3
-----Records Updated -----
```

```
mysql> select * from student;
```

ID	NAME	AGE
1	abc	20
2	Nam	20
3	Nam	20
4	sam	25
5	rat	200
6	mat	22
8	Nuha	2
9	Ayan	15
10	Dog	13

```
9 rows in set (0.02 sec)
```

```
mysql> select * from student;
```

ID	NAME	AGE
1	abc	20
3	Nam	23
4	sam	25
5	rat	200
6	mat	22
8	Nuha	2
9	Ayan	15
10	Dog	13
11	King	45
12	Queen	42
13	Worker	23

```
11 rows in set (0.01 sec)
```

```
mysql>
```

### SQL Code for creating Table

```
CREATE TABLE Student(
ID INT NOT NULL AUTO_INCREMENT, NAME VARCHAR(20) NOT NULL,
AGE INT NOT NULL, PRIMARY KEY (ID)
);
```

## **B. Write a program to demonstrate PreparedStatement in Spring JdbcTemplate**

Code:

### **SQL Code for creating Table**

```
CREATE TABLE Student(  
ID INT NOT NULL AUTO_INCREMENT, NAME VARCHAR(20) NOT NULL,  
AGE INT NOT NULL, PRIMARY KEY (ID)  
);
```

### **MvSpringJDBC.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-3.0.xsd ">  
  
<!-- Initialization for data source -->  
<bean id="dataSource"  
class = "org.springframework.jdbc.datasource.DriverManagerDataSource">  
<property name = "driverClassName" value = "com.mysql.jdbc.Driver"/>  
<property name = "url" value = "jdbc:mysql://localhost:3306/imcost?useSSL=false"/>  
<property name = "username" value = "root"/>  
<property name = "password" value = "1234"/>  
</bean>  
  
<!-- Definition for studentJdbcTemplate bean -->  
<bean id = "studentJdbcTemplate"  
class = "com.test.StudentJdbcTemplate">  
<property name = "dataSource" ref = "dataSource" />  
</bean>  
</beans>
```

### **StudentDAO.java**

```
package com.test; import java.util.List;  
import javax.sql.DataSource;  
import java.util.List;  
import javax.sql.DataSource;  
  
public interface StudentDAO  
{  
public void setDataSource(DataSource ds);  
public Student getStudent(Integer id);  
}
```

### **Student.java**

```
package com.test;

public class Student
{
    private Integer age; private String name;
    private Integer id;

    public void setAge(Integer age)
    {
        this.age = age;
    }

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

### **StudentJdbcTemplate.java**

```
package com.test;

import java.sql.PreparedStatement; import java.sql.SQLException; import java.util.List;
import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementSetter;
```

```

public class StudentJDBCTemplate implements StudentDAO {
private DataSource dataSource;
private JdbcTemplate jdbcTemplateObject;
public void setDataSource(DataSource dataSource) {
this.dataSource = dataSource;
this.jdbcTemplateObject = new JdbcTemplate(dataSource);
}
public Student getStudent(final Integer id) {
final String SQL = "select * from Student where id = ? ";
List <Student> students = jdbcTemplateObject.query(
SQL, new PreparedStatementSetter() {

public void setValues(PreparedStatement preparedStatement) throws
SQLException {
preparedStatement.setInt(1, id);
}
},
new StudentMapper());
return students.get(0);
}
}

```

### **MyMain.java**

```

package com.test;
import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MyMain {
public static void main(String[] args) { ApplicationContext context = new
ClassPathXmlApplicationContext("MySpringJDBC.xml"); StudentJDBCTemplate
studentJDBCTemplate =
(StudentJDBCTemplate)context.getBean("studentJDBCTemplate");
Student student = studentJDBCTemplate.getStudent(5);
System.out.print("ID : " + student.getId() );
System.out.println(", Age : " + student.getAge());
}
}

```

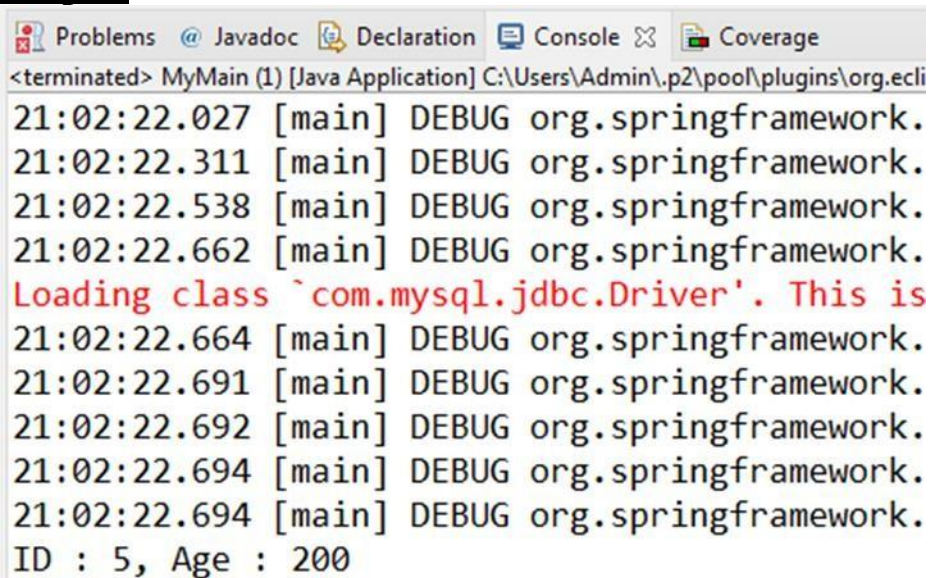


## StudentMapper.java

```
package com.test;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

public class StudentMapper implements RowMapper<Student> {
    public Student mapRow(ResultSet rs, int rowNum) throws SQLException {
        Student student = new Student();
        student.setId(rs.getInt("id"));
        student.setName(rs.getString("name"));
        student.setAge(rs.getInt("age"));
        return student;
    }
}
```

## Output:



```
<terminated> MyMain (1) [Java Application] C:\Users\Admin\p2\pool\plugins\org.ecli
21:02:22.027 [main] DEBUG org.springframework.
21:02:22.311 [main] DEBUG org.springframework.
21:02:22.538 [main] DEBUG org.springframework.
21:02:22.662 [main] DEBUG org.springframework.
Loading class `com.mysql.jdbc.Driver`. This is
21:02:22.664 [main] DEBUG org.springframework.
21:02:22.691 [main] DEBUG org.springframework.
21:02:22.692 [main] DEBUG org.springframework.
21:02:22.694 [main] DEBUG org.springframework.
21:02:22.694 [main] DEBUG org.springframework.
ID : 5, Age : 200
```

## C. Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface

Code:

### **SQL Code for creating Table**

```
CREATE TABLE Student(
ID INT NOT NULL AUTO_INCREMENT, NAME VARCHAR(20) NOT NULL,
AGE INT NOT NULL, PRIMARY KEY (ID)
);
```



### **MySpringJDBC.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-3.0.xsd ">

<!-- Initialization for data source -->
<bean id="dataSource"
class = "org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name = "driverClassName" value = "com.mysql.jdbc.Driver"/>
<property name = "url" value = "jdbc:mysql://localhost:3306/imcost?useSSL=false"/>
<property name = "username" value = "root"/>
<property name = "password" value = "1234"/>
</bean>

<!-- Definition for studentJdbcTemplate bean -->
<bean id = "studentJdbcTemplate"
class = "com.test.StudentJdbcTemplate">
<property name = "dataSource" ref = "dataSource" />
</bean>
</beans>
```

### **StudentDAO.java**

```
package com.test;
import java.util.List;
import javax.sql.DataSource;

public interface StudentDAO {

public void setDataSource(DataSource ds);

public List<Student> listStudents();
}
```

### **Student.java**

```
package com.test;
public class Student
{
private Integer age; private String name; private Integer id;

public void setAge(Integer age)
{
this.age = age;
}
```

```

public Integer getAge()
{
    return age;
}
public void setName(String name)
{
    this.name = name;
}
public String getName()
{
    return name;
}
public void setId(Integer id)
{
    this.id = id;
}
public Integer getId()
{
    return id;
}
}

```

### **StudentJdbcTemplate.java**

```

package com.test;
import java.util.List;
import java.sql.ResultSet; import java.sql.SQLException; import java.util.ArrayList;
import javax.sql.DataSource;

```

```

import org.springframework.dao.DataAccessException; import
org.springframework.jdbc.core.JdbcTemplate; import
org.springframework.jdbc.core.ResultSetExtractor;

```

```

public class StudentJdbcTemplate implements StudentDAO {
    private DataSource dataSource;
    private JdbcTemplate jdbcTemplateObject;

```

```

    public void setDataSource(DataSource dataSource) {

```

```

        this.dataSource = dataSource;
        this.jdbcTemplateObject = new JdbcTemplate(dataSource);
    }
    public List<Student> listStudents() { String SQL = "select * from Student";
    List <Student> students = jdbcTemplateObject.query(SQL,
    new ResultSetExtractor<List<Student>>(){

```

```
public List<Student> extractData(  
ResultSet rs) throws SQLException, DataAccessException {
```

```
List<Student> list = new ArrayList<Student>();
```

```
while(rs.next()){
```

```
Student student = new Student();
```

```
student.setId(rs.getInt("id"));
```

```
student.setName(rs.getString("name"));
```

```
student.setAge(rs.getInt("age"));
```

```
list.add(student);
```

```
}
```

```
return list;
```

```
}
```

```
});
```

```
return students;
```

```
}
```

```
}
```

### **MyMain.java**

```
package com.test;
```

```
import java.util.List;
```

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

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

```
public class MyMain
```

```
{
```

```
public static void main(String[] args)
```

```
{
```

```
ApplicationContext context = new
```

```
ClassPathXmlApplicationContext("MySpringJDBC.xml");
```

```
StudentJDBCTemplate studentJDBCTemplate =
```

```
(StudentJDBCTemplate)context.getBean("studentJDBCTemplate");
```

```
List<Student> students = studentJDBCTemplate.listStudents();
```

```
for(Student student: students)
```

```
{
```

```
System.out.print("ID : " + student.getId() );
```

```
System.out.println(", Age : " + student.getAge());
```

```
}
```

```
}
```

## Output:

```
Problems Javadoc Declaration Console Coverage
<terminated> MyMain (1) [Java Application] C:\Users\Admin\p2\pool\plugin
21:17:38.896 [main] DEBUG org.springframework
21:17:38.969 [main] DEBUG org.springframework
21:17:39.061 [main] DEBUG org.springframework
21:17:39.062 [main] DEBUG org.springframework
21:17:39.083 [main] DEBUG org.springframework
21:17:39.086 [main] DEBUG org.springframework
21:17:39.086 [main] DEBUG org.springframework
Loading class `com.mysql.jdbc.Driver'. Th
ID : 1, Age : 20
ID : 3, Age : 23
ID : 4, Age : 25
ID : 5, Age : 200
ID : 6, Age : 22
ID : 8, Age : 2
ID : 9, Age : 15
ID : 10, Age : 13
ID : 11, Age : 45
ID : 12, Age : 42
ID : 13, Age : 23

21:32:36.698 [main] DEBUG org.springframework.jdbc
21:32:36.699 [main] DEBUG org.springframework.bean
-----Listing Multiple Records-----
21:32:36.729 [main] DEBUG org.springframework.jdbc
21:32:36.732 [main] DEBUG org.springframework.jdbc
21:32:36.732 [main] DEBUG org.springframework.jdbc
ID : 1, Name : abc, Age : 20
ID : 3, Name : Nam, Age : 23
ID : 4, Name : sam, Age : 25
ID : 5, Name : rat, Age : 200
ID : 6, Name : mat, Age : 22
ID : 8, Name : Nuha, Age : 2
ID : 9, Name : Ayan, Age : 15
ID : 10, Name : Dog, Age : 13
ID : 11, Name : King, Age : 45
ID : 12, Name : Queen, Age : 42
ID : 13, Name : Worker, Age : 23
```

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

Code:

**SQL Code for creating Table**

```
CREATE TABLE Student(  
ID INT NOT NULL AUTO_INCREMENT, NAME VARCHAR(20) NOT NULL,  
AGE INT NOT NULL, PRIMARY KEY (ID)  
);
```

**MySpringJDBC.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-3.0.xsd ">  
<!-- Initialization for data source -->  
<bean id="dataSource"  
class = "org.springframework.jdbc.datasource.DriverManagerDataSource">  
<property name = "driverClassName" value = "com.mysql.jdbc.Driver"/>  
<property name = "url" value = "jdbc:mysql://localhost:3306/imcost?useSSL=false"/>  
<property name = "username" value = "root"/>  
<property name = "password" value = "1234"/>  
</bean>  
<!-- Definition for studentJDBCTemplate bean -->  
<bean id = "studentJDBCTemplate"  
class = "com.test.StudentJDBCTemplate">  
<property name = "dataSource" ref = "dataSource" />  
</bean>  
</beans>
```

**StudentDAO.java**

```
package com.test;  
import java.util.List;  
import javax.sql.DataSource;  
  
public interface StudentDAO {  
public void setDataSource(DataSource ds);  
public List<Student> listStudents();  
}
```

```
Student.java package com.test;  
public class Student  
{
```

```

private Integer age; private String name; private Integer id;
public void setAge(Integer age)
{
    this.age = age;
}
public Integer getAge()
{
    return age;
}
public void setName(String name)
{
    this.name = name;
}
public String getName()
{
    return name;
}
public void setId(Integer id)
{
    this.id = id;
}
public Integer getId()
{
    return id;
}
}

```

### **StudentJDBCTemplate.java**

```

package com.test;
import java.util.List;
import javax.sql.DataSource;
import org.springframework.jdbc.core.JdbcTemplate;

public class StudentJDBCTemplate implements StudentDAO
{
    private DataSource dataSource;
    private JdbcTemplate jdbcTemplateObject;
    public void setDataSource(DataSource dataSource)
    {
        this.dataSource = dataSource;
        this.jdbcTemplateObject = new JdbcTemplate(dataSource);
    }
    public List<Student> listStudents()

```

```

{
String SQL = "select * from Student";
List <Student> students = jdbcTemplateObject.query(SQL, new StudentMapper());
return students;
}
}

```

### **MyMain.java**

```

package com.test;
import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MyMain
{
public static void main(String[] args)
{
ApplicationContext context = new
ClassPathXmlApplicationContext("MySpringJDBC.xml"); StudentJdbcTemplate
studentJdbcTemplate =
(StudentJdbcTemplate)context.getBean("studentJdbcTemplate");

System.out.println("-----Listing Multiple Records ");
List<Student> students = studentJdbcTemplate.listStudents();

for (Student record : students)
{
System.out.print("ID : " + record.getId() ); System.out.print(", Name : " +
record.getName() ); System.out.println(", Age : " + record.getAge());
}
}
}

```

### **StudentMapper.java**

```

package com.test;
import java.sql.ResultSet;

import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;

public class StudentMapper implements RowMapper<Student>
{
public Student mapRow(ResultSet rs, int rowNum) throws SQLException
{

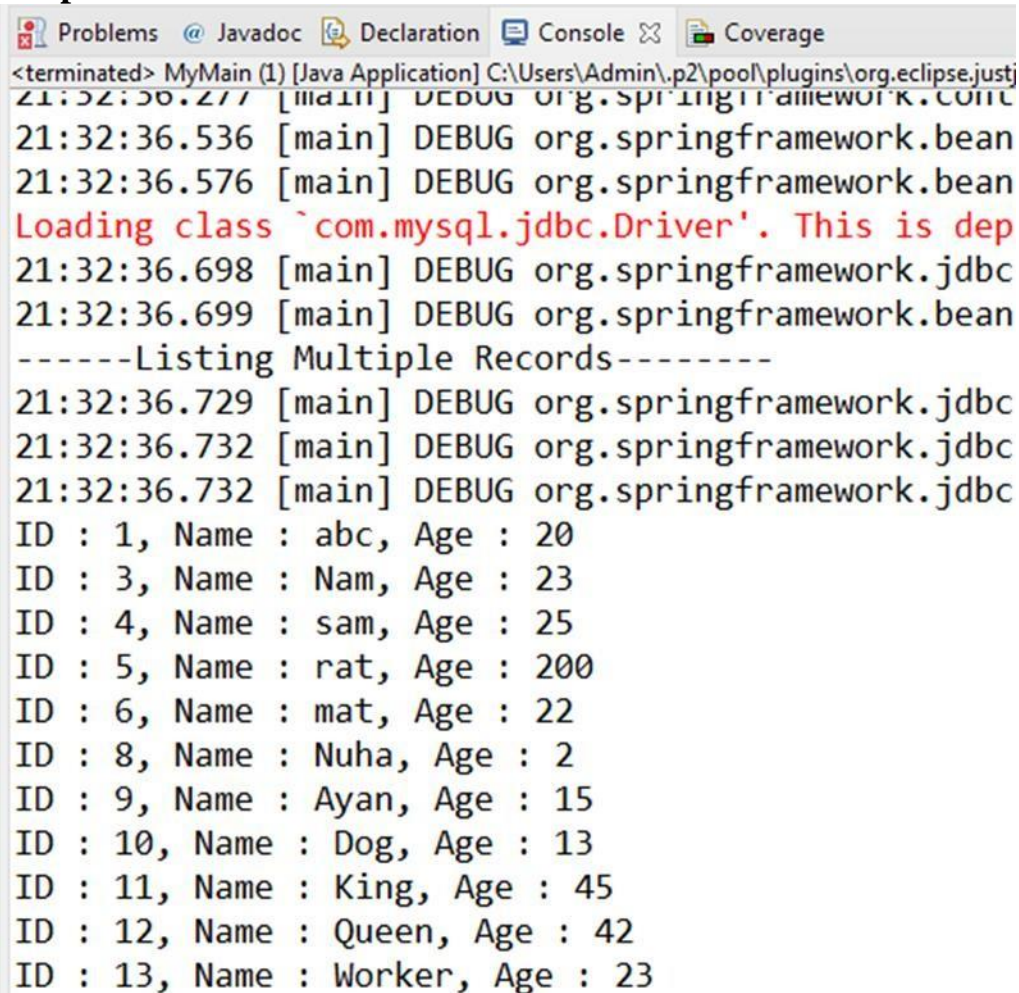
```

```

Student student = new Student(); student.setId(rs.getInt("id"));
student.setName(rs.getString("name")); student.setAge(rs.getInt("age")); return student;
}
}

```

## Output:



The screenshot shows the Eclipse IDE console with the following output:

```

<terminated> MyMain (1) [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.just
21:32:36.277 [main] DEBUG org.springframework.conc
21:32:36.536 [main] DEBUG org.springframework.bean
21:32:36.576 [main] DEBUG org.springframework.bean
Loading class `com.mysql.jdbc.Driver'. This is dep
21:32:36.698 [main] DEBUG org.springframework.jdbc
21:32:36.699 [main] DEBUG org.springframework.bean
-----Listing Multiple Records-----
21:32:36.729 [main] DEBUG org.springframework.jdbc
21:32:36.732 [main] DEBUG org.springframework.jdbc
21:32:36.732 [main] DEBUG org.springframework.jdbc
ID : 1, Name : abc, Age : 20
ID : 3, Name : Nam, Age : 23
ID : 4, Name : sam, Age : 25
ID : 5, Name : rat, Age : 200
ID : 6, Name : mat, Age : 22
ID : 8, Name : Nuha, Age : 2
ID : 9, Name : Ayan, Age : 15
ID : 10, Name : Dog, Age : 13
ID : 11, Name : King, Age : 45
ID : 12, Name : Queen, Age : 42
ID : 13, Name : Worker, Age : 23

```



## Practical No 10

### Aim: Assignment based Spring Boot and RESTful Web Services

#### Description:

REST stands for **Representational State Transfer**. It is developed by **Roy Thomas Fielding**, who also developed HTTP. The main goal of RESTful web services is to make web services **more effective**. RESTful web services try to define services using the different concepts that are already present in HTTP. REST is an **architectural approach**, not a protocol.

It does not define the standard message exchange format. We can build REST services with both XML and JSON. JSON is more popular format with REST. The **key abstraction** is a resource in REST. A resource can be anything. It can be accessed through a **Uniform Resource Identifier (URI)**.

**A. Write a program to create a simple Spring Boot application that prints a message.**

Code:

#### pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.test</groupId>
<artifactId>MondayBoot</artifactId>
<version>0.0.1-SNAPSHOT</version>

<description>spring boot</description>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.0.3.RELEASE</version>
</parent>
<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
<version>2.2.2.RELEASE</version>
</dependency>
```

```

</dependencies>

<properties>
<java.version>1.8</java.version>
</properties>
<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>
</project>

```

### **MyMain.java**

```

package com.abc;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication; import
org.springframework.context.ApplicationContext; @SpringBootApplication
public class MyMain
{
public static void main(String[] args)
{
ApplicationContext con= SpringApplication.run(MyMain.class, args);
}
}

```

### **MyController.java**

```

package com.abc;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController; @RestController
public class MyController
{
@RequestMapping("/Home")
public String welcome()
{
return "welcome to springBoot";
}
}

```

### **Output:**



welcome to springBoot

## **B. Write a program to demonstrate RESTful Web Services with spring boot.**

Code:

### **pom.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.4.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>net.codejava</groupId>
  <artifactId>HelloREST</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>HelloREST</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>

    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
      </plugin>
```

```
</plugins>
</build>
</project>
```

### **HelloRestApplication.java**

```
package net.codejava.ws;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication

public class HelloRestApplication {
    public static void main(String[] args) { SpringApplication.run(HelloRestApplication.class,
args);
    }
}
```

### **HelloController.java**

```
package net.codejava.ws;
import org.springframework.web.bind.annotation.RequestMapping; import
org.springframework.web.bind.annotation.RequestParam; import
org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloController
{
    @RequestMapping("/hello")
    public String hello(@RequestParam(name = "name", defaultValue = "Earth") String
name)
    {
        return "Hello World RESTFull with Spring boot Name:" + name ;
    }
}
```

### **Output:**

