BharatiVidyapeeth's

Institute of Management & InformationTechnology

C.B.D. Belapur, NaviMumbai 400614

Vision:
Providing high quality, innovative and value-based education in information technology to build competent professionals.

Mission:
M1. To provide students for career in the industry and pursue research in all branches of computing field through effective teaching learning process.
M2. To strengthen the industry institute interaction by providing up-to-data programs.
M3. To embibe amongst the students ethical usage of technical knowledge beneficial to society. M4. To provide an environment that fosters a framework for promoting collaborative and multidisciplinary activities.

CERTIFICATE

This is to certify that the journal is the work of **Mr. Sainath Ravi Machha** Roll No.**32** of **MCA** (**Sem-I** Div: **A**) for the academic year **2023-2024.**
Subject Code: **MCAL12**
Subject Name: **Advanced Java Lab**

Subject-in-charge

Principal

Date: _____

External Examiner

Bharati Vidyapeeth's Institute of Management & Information Technology MCA Semester I AY2023-24

MCAL12: Advanced Java

INDEX

Name: Sainath Ravi Machha **MCA SEM-1 Division: A Batch: A2**

CO1 Demonstrate use of data structure and data manipulation concept using Java Collection Framework and Lambda expressions.
CO2 Build JSP web application using standard actions, custom tags and JSTL Tags.
CO3 Develop application using Spring Framework, Lightweight Containers and Dependency Injection with Spring.
CO4 Develop applications using Aspect Oriented Programming with Spring CO5 Build JDBC application with Spring using JdbcTemplate.
CO6 Develop Spring Boot Web Application and Spring Boot RESTful web services.

Sr. No.	Date	Topic	CO	CO-PO MAP	Attentiveness(5)	Presentation (05)	Understanding (10)	Attendance (25)	Total(25)	Sign
1		**Java Generics**	**CO1**	**PO1,PO2, PO3,PO4, PO5, PO9,PSO1 ,PSO2**						
1.1	29.8.23	Write a Java Program to demonstrate a Generic Class.								
1.2	29.8.23	Write a Java Program to demonstrate Generic Methods.								

1.3	29.8.23	Write a Java Program to demonstrate Wildcards in Java Generics.								
2		**List Interface**	**CO1**	**PO1,PO2, PO3,PO4, PO5, PO9,PSO1 ,PSO2**						
2.1	26.9.23	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.								
2.2	26.9.23	Write a Java program to create List containing list of items and use ListIterator interface to print items present								
3		**Set Interface**	**CO1**	**PO1,PO2, PO3,PO4, PO5, PO9,PSO1 ,PSO2**						

3.1	3.10.23	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.								
	3.10.23	Write a Java program using Set interface containing list of items and perform the following operations:								
		a. Add items in the set. b. b. Insert items of one set in to other set. c. c. Remove items from the set								
4		**Map Interface**	**CO1**	**PO1,PO2, PO3,PO4, PO5, PO9,PSO1 ,PSO2**						

4.1	3.10.23	1. Remove items from the map 2. Search specific key from the map 3. Get value of the specified key 4. Insert map elements of one map in to other map. 5. Print all keys and values of the map.								
5		**Lambda Expression**	**CO1**	**PO1,PO2, PO3,PO4, PO5, PO9,PSO1 ,PSO2**						
5.1	26.9.23	Write a Java program using Lambda Expression to print ”Hello World”.								
5.2	26.9.23	Write a Java program using Lambda Expression with single parameter.								
5.3	26.9.23	Write a Java program using Lambda Expression with multiple parameters to add two numbers.								

5.4	26.9.23	Write a Java program using Lambda Expression to calculate the following: a. Convert Fahrenheit to Celcius b. Convert Kilometers to Miles.								
5.5	26.9.23	Write a Java program using Lambda Expression with or without return keyword.								
5.6	26.9.23	Write a Java program using Lambda Expression to concatenate two strings.								
6		**Web application development using JSP**	**CO2**	**PO1,PO2, PO3,PO4, PO5, PO9,PSO1 ,PSO2**						

6.1	6.10.23	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.								
6.2	6.10.23	Write a JSP page to display the Registration form (Make your own assumptions).								
6.3	6.10.23	Write a JSP program to add, delete and display the recordsfrom StudentMaster (RollNo, Name, Semester, Course) table.								
6.4	10.10.23	Design loan calculator using JSP which accepts Period ofTime (in years) and Principal Loan Amount. Display thepayment amount for each loan and then list the loanbalance and interest paid for each payment over the								

		term ofthe loan for the following time								
		period and interest rate: a. 1 to 7 year at 5.35% b. 8 to 15 year at 5.5% c. 16 to 30 year at 5.75%								
6.5	31.10.23	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.								

6.6	31.10.23	Write a JSP program to add, delete and display the records from StudentMaster (RollNo, Name, Semester, Course) table.								
6.7	17.10.23	Write a JSP program that demonstrates the use of JSP declaration, scriptlet,directives, expression,header and footer.								
6.8	17.10.23	Write a JSP program that demonstrates the use of Cookies and session tracking in java.								
7		**Spring Framework**	**CO3**	**PO1,PO2, PO3,PO4, PO5, PO9,PSO1 ,PSO2**						
7.1	6.11.23	Write a program to print “Hello World” using spring framework.								
7.2	6.11.23	Write a program to demonstrate dependency injection via setter method.								

7.3	6.11.23	Write a program to demonstrate dependency injection via Constructor.								
8		**Aspect Oriented Programming**	**CO4**	**PO1,PO2, PO3, PO4,PO5, PO7, PO9, PO11,PSO 1,PSO2**						
8.1	7.11.23	Write a program to demonstrate Spring AOP – before advice								
8.2	7.11.23	Write a program to demonstrate Spring AOP – after advice.								
8.3	7.11.23	Write a program to demonstrate Spring AOP – around advice								
8.4	7.11.23	Write a program to demonstrate Spring AOP – after returning advice								
8.5	7.11.23	Write a program to demonstrate Spring AOP – after throwing advice.								
9		**Spring JDBC**	**CO5**	**PO1,PO2, PO3, PO4,PO5, PO7, PO9, PO11,PSO 1,PSO2**						

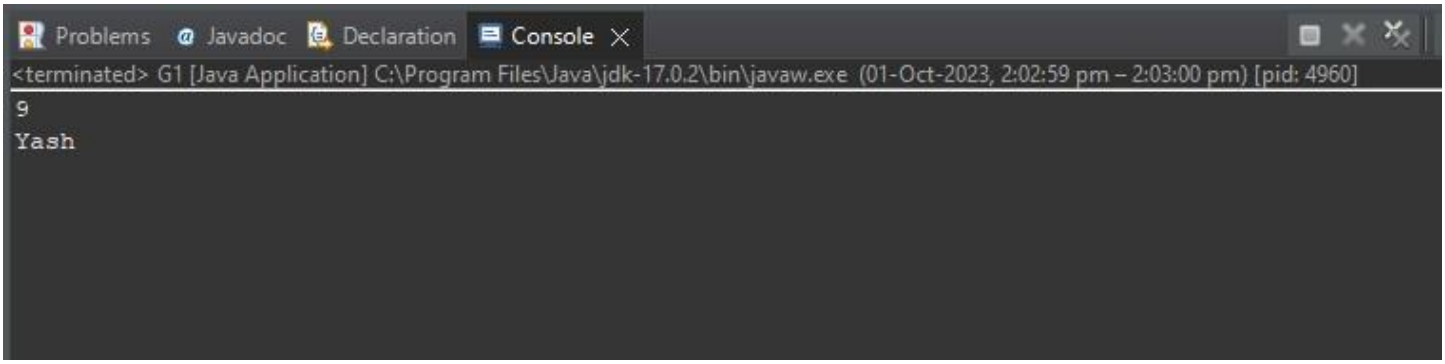
9.1	29.11.23	Write a program to insert, update and delete records from given table								
9.2	29.11.23	Write a program to demonstrate PreparedStatement in Spring JdbcTemplate.								
9.3	29.11.23	Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface.								
9.4	29.11.23	Write a program to demonstrate RowMapper interface to fetch the records from the database.								
10		**Spring Boot and RESTful Web Services**	**CO6**	**PO1,PO2, PO3, PO4,PO5, PO7, PO9, PO11,PSO 1,PSO2**						
10.1	6.12.23	Write a program to create a simple Spring Boot application that prints a message.								
10.2	6.10.23	Write a program to demonstrate RESTful Web Services with spring boot.								

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

Code :

```
class geg<T>
{
   T obj;
   geg(T obj){this.obj = obj;}
public T get() {returnthis.obj;}
}
class G1
{
publicstaticvoid main (String[] args)
   {
      geg<Integer>i=new geg<Integer>(9);
      System.out.println(i.get());

      geg<String> s =
new geg<String>("Yash ");
      System.out.println(s.get());
   }
}
```

Output :

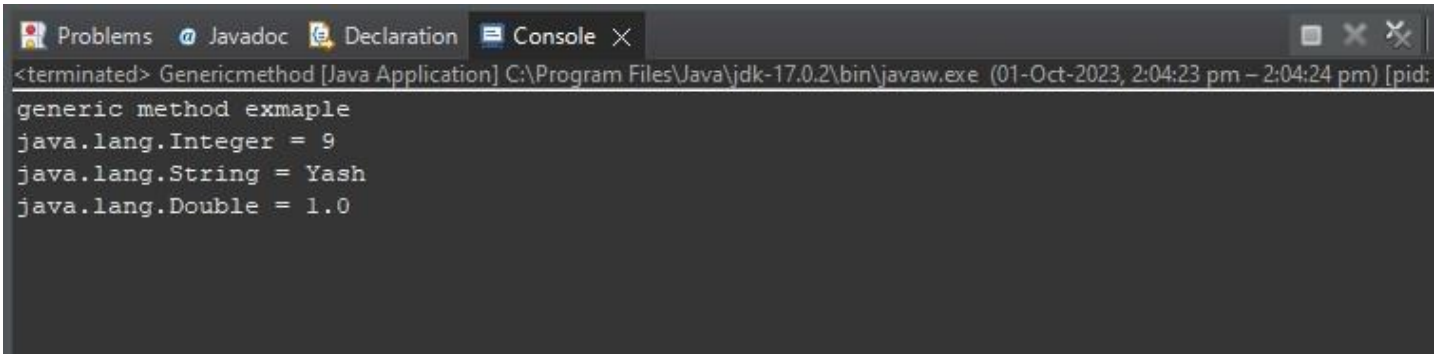


Problem Statement 2 : Write a Java Program to demonstrate Generic Methods.

Code :

```
publicclass Genericmethod
{
void display()
   {
      System.out.println("generic method exmaple");
   }
<T>void gdisplay (T e)
   {
      System.out.println(e.getClass().getName() + " = " + e);
   }
publicstaticvoid main(String[] args)
   {
      Genericmethod g1=new Genericmethod();
      g1.display();

      g1.gdisplay(9);
      g1.gdisplay("Yash");
      g1.gdisplay(1.0);
   }
   }
```

Output :

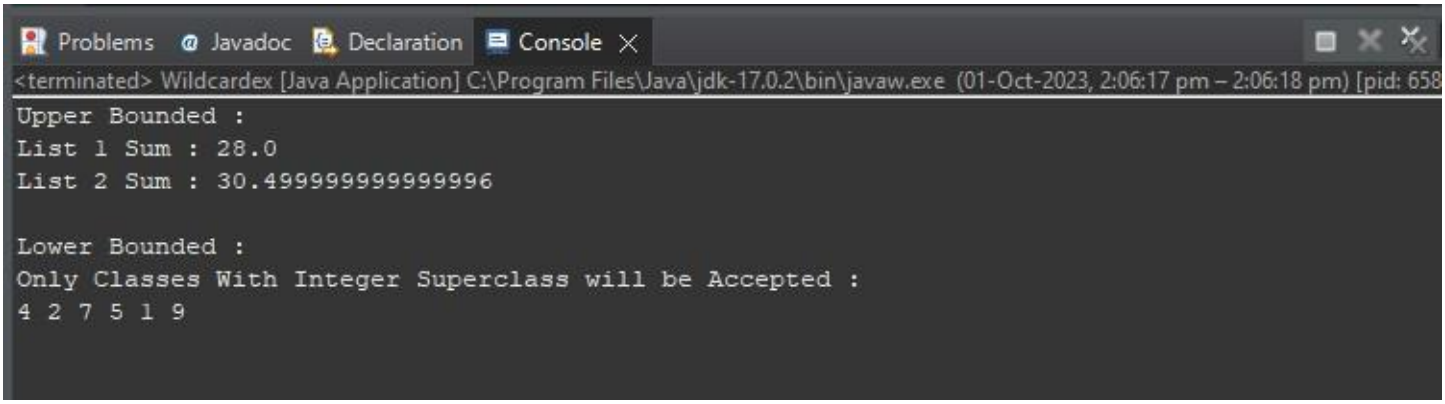


Problem Statement 3 : Write a Java Program to demonstrate Wildcards in Java Generics.

Code :

```
import java.util.*;
public class Wildcardex {
   // Upper bounded
   private static double sum(List<? extends Number> list) {
      double sum = 0.0;
      for (Number i : list) {
         sum = sum + i.doubleValue();
      }
      return sum;
   }
   // Lower Bounded
   private static void show(List<? super Integer> list) {
      list.forEach((x) -> {
 System.out.print(x + "");
      });
   }
   public static void main(String[] args) {
      System.out.println("Upper Bounded : ");
      List<Integer> list1 = Arrays.asList(4, 2, 7, 5, 1, 9);
      System.out.println("List 1 Sum : " + sum(list1));
      List<Double> list2 = Arrays.asList(4.7, 2.4, 7.3, 5.4, 1.5, 9.2);
      System.out.println("List 2 Sum : " + sum(list2));
      System.out.println("\nLower Bounded : ");
      List<Integer> list3 = Arrays.asList(4, 2, 7, 5, 1, 9);
      System.out.println("Only Classes With Integer Superclass will be Accepted : ");
      show(list3);
   }
}
```

Output :

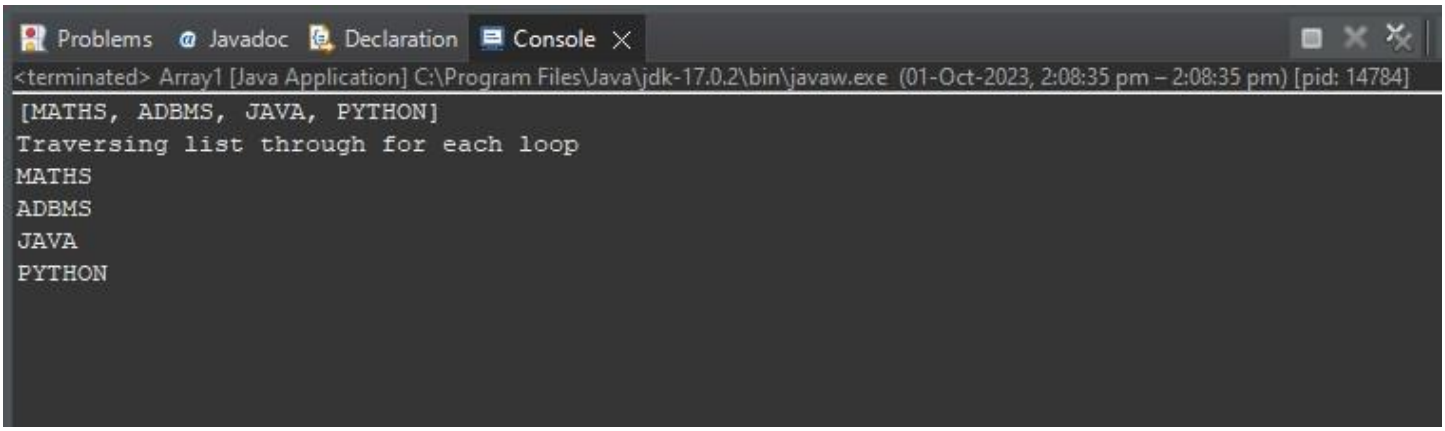

Problems
Javadoc
Declaration
Console
<terminated> Wildcardex [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (01-Oct-2023, 2:06:17 pm – 2:06:18 pm) [pid: 658
Upper Bounded :
List 1 Sum : 28.0
List 2 Sum : 30.499999999999996
Lower Bounded :
Only Classes With Integer Superclass will be Accepted :
4 2 7 5 1 9

Problem Statement 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.

Code :

```
package listeg;
import java.util.*;
public class Array1 {
	public static void main(String[] args) {
		ArrayList<String>list=new ArrayList<String>();

		list.add("MATHS");
		list.add("ADBMS");
		list.add("JAVA");
		list.add("PYTHON");

		System.out.println(list);

		System.out.println("Traversing list through for each loop");
		for(String subject:list)
			System.out.println(subject);

	}

}
```

Output :

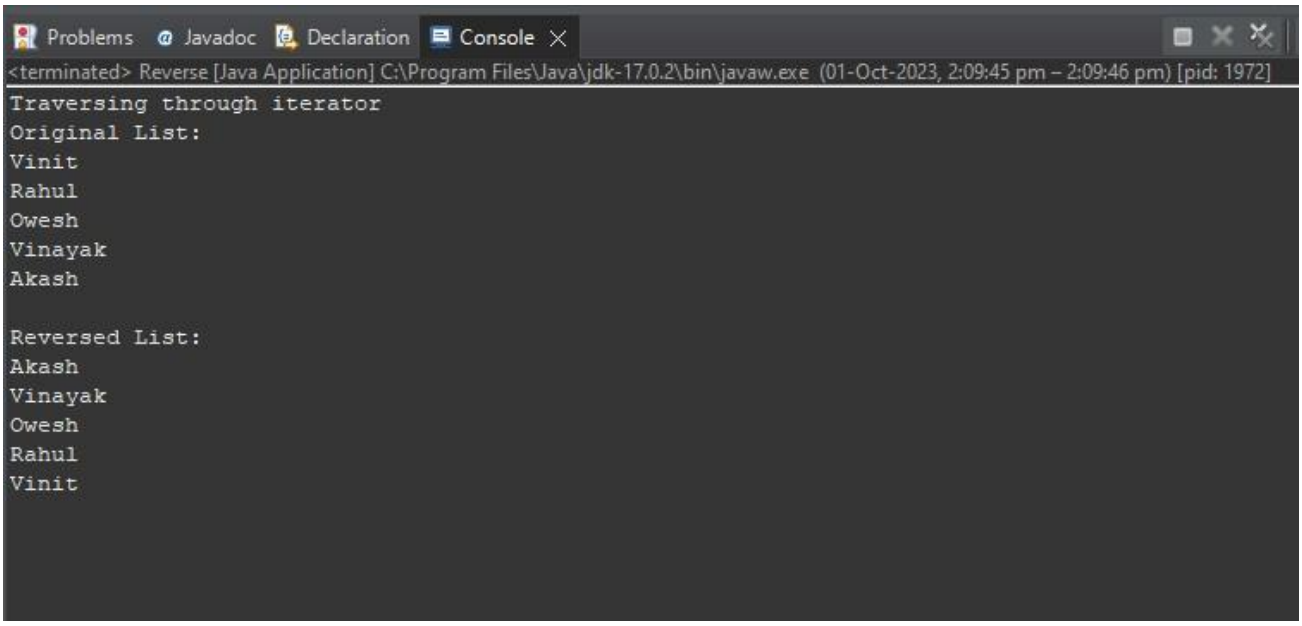


Problem Statement 2 : Write a Java program to create List containing list of items and use ListIterator interface to print items present in the list. Also print the list in reverse/ backward direction.

Code :

```
package listeg;
import java.util.*;
public class Reverse {

	public static void main(String[] args) {
		List<String> mylist = new ArrayList<String>();

		mylist.add("Vinit");
		mylist.add("Rahul");
		mylist.add("Owesh");
		mylist.add("Vinayak");
		mylist.add("Akash");

		System.out.println("Traversing through iterator");
		System.out.println("Original List:");
		Iterator itr=mylist.iterator();
		while(itr.hasNext()) {
			System.out.println(itr.next());
			}
		Collections.reverse(mylist);
		System.out.println(); //space between two lines
		System.out.println("Reversed List:");
		Iterator itr1=mylist.iterator();
		while(itr1.hasNext()) {

			System.out.println(itr1.next());
		}
}
}
```

Output :

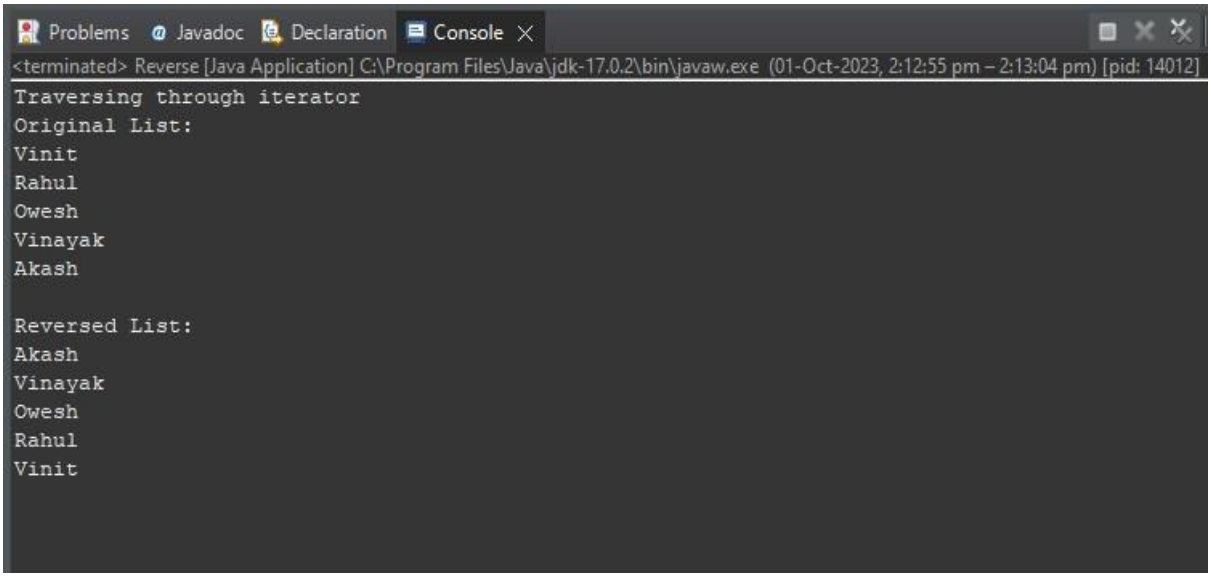

Problems
Javadoc
Declaration
Console
<terminated> Reverse [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (01-Oct-2023, 2:09:45 pm – 2:09:46 pm) [pid: 1972]
Traversing through iterator
Original List:
Vinit
Rahul
Owesh
Vinayak
Akash
Reversed List:
Akash
Vinayak
Owesh
Rahul
Vinit

Problem Statement 1 : 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/ backword direction.

Solution :

```
import java.util.*;
publicclass Reverse {
publicstaticvoid main(String[] args) {
// Let us create a list of strings
List<String> mylist = new ArrayList<String>();
mylist.add("vinit");
mylist.add("owesh");
mylist.add("sudarshan");
mylist.add("sushant");
System.out.println("Original list ");

Iterator<String> itr=mylist.iterator();//getting the Iterator
while(itr.hasNext()){//check if iterator has the elements
System.out.println(itr.next());
}
Collections.reverse(mylist);
System.out.println("");
System.out.println("reversed list ");

Iterator<String> itr1=mylist.iterator();//getting the Iterator
while(itr1.hasNext()){//check if iterator has the elements

System.out.println(itr1.next());
}
}
}
```

Output :



Problem Statement2 : Write a Java program using Set interface containing list of items and perform the following operations:

a. Add items in the set.

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

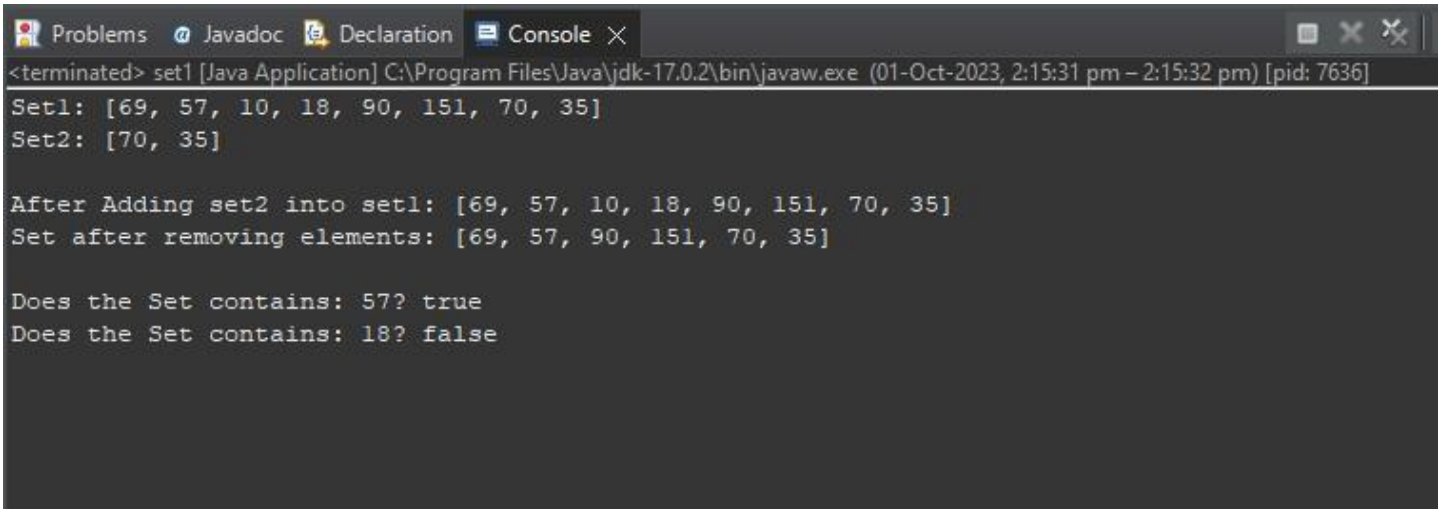
c. Remove items from the set

d. Search the specified item in the set

Solution :

```
import java.util.*;
publicclass set1{
publicstaticvoid main(String[] args) {
// TODO Auto-generated method stub
 Set<Integer> s = new LinkedHashSet<Integer>();
 s.add(69);
 s.add(57);
 s.add(10);
 s.add(18);
 s.add(90);
 s.add(151);
 Set<Integer> s1 = new LinkedHashSet<Integer>();
 s1.add(70);
 s1.add(35);

 s.addAll(s1);
 System.out.println("Set1: " + s);
 System.out.println("Set2: " + s1);
 System.out.println();
 System.out.println("After Adding set2 into set1: " + s);
 s.remove(10);
 s.remove(18);
 System.out.println("Set after removing elements: " + s);
 System.out.println();
 System.out.println("Does the Set contains: 57? "
 + s.contains(57));
 System.out.println("Does the Set contains: 18? "
 + s.contains(18));
}
}
```

Output :


Problems
Javadoc
Declaration
Console
<terminated> set1 [Java Application] C:\Program Files\Java\jdk-17.0.2\bin\javaw.exe (01-Oct-2023, 2:15:31 pm – 2:15:32 pm) [pid: 7636]
Set1: [69, 57, 10, 18, 90, 151, 70, 35]
Set2: [70, 35]
After Adding set2 into set1: [69, 57, 10, 18, 90, 151, 70, 35]
Set after removing elements: [69, 57, 90, 151, 70, 35]
Does the Set contains: 57? true
Does the Set contains: 18? false

Map Interface

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

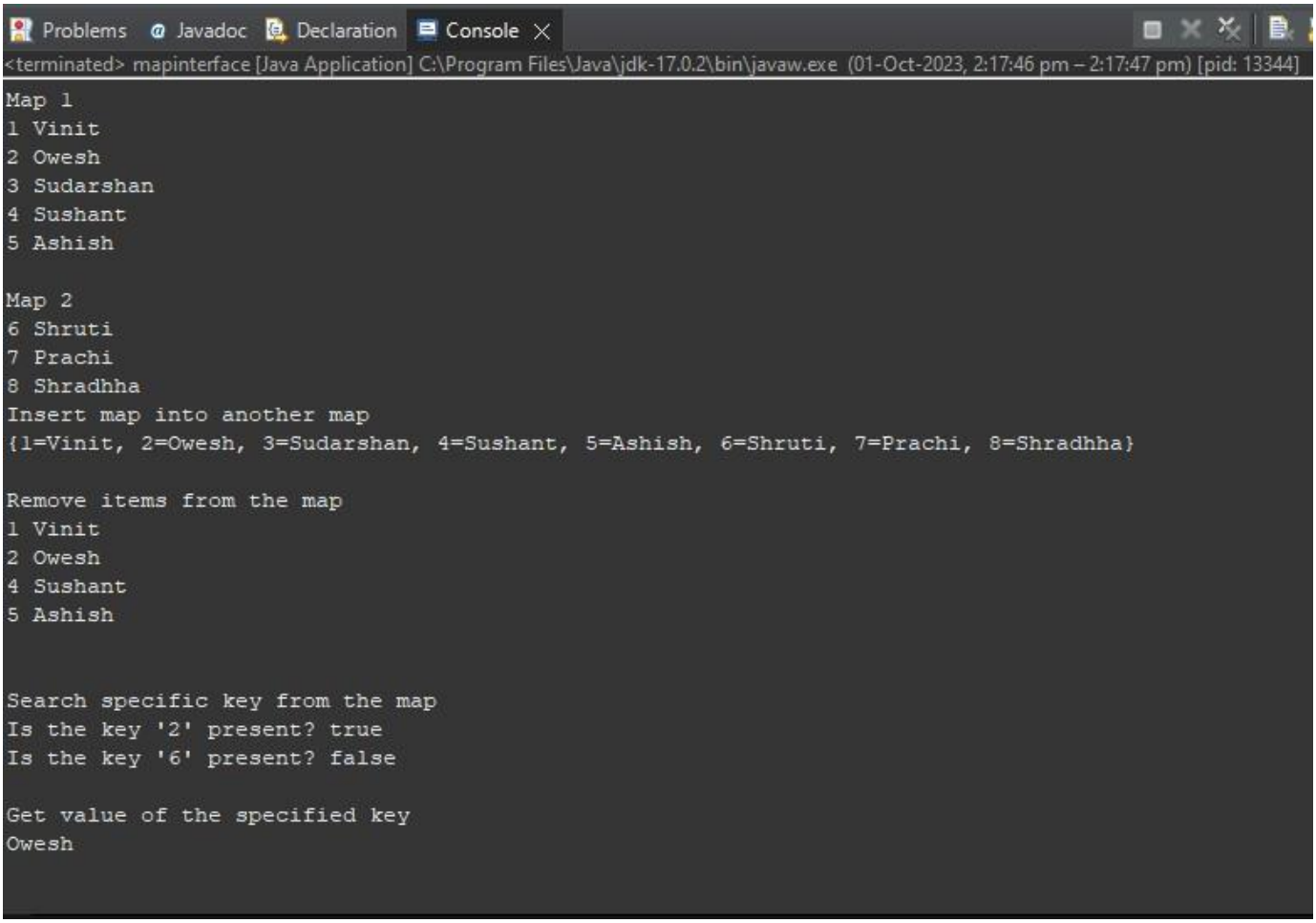
a. Add items in the map.
b. Remove items from the map
c. Search specific key from the map
d. Get value of the specified key
e. Insert map elements of one map in to other map.
f. Print all keys and values of the map.

Solution :

```
import java.util.*;
publicclass mapinterface {
publicstaticvoid main(String[] args) {
// TODO Auto-generated method stub
Map<Integer, String> map = new HashMap<>();
map.put(1 ,"Vinit");
map.put(2,"Owesh");
map.put(3,"Sudarshan");
map.put(4,"Sushant");
map.put(5,"Ashish");
System.out.println();
Map<Integer, String> map1 = new HashMap<>();
map1.put(6 ,"Shruti");
map1.put(7,"Prachi");
map1.put(8,"Shradhha");
System.out.println("Map 1");
for (Map.Entry<Integer, String> e : map.entrySet())
System.out.println(e.getKey() + "" + e.getValue());
System.out.println();
System.out.println("Map 2");
for (Map.Entry<Integer, String> e : map1.entrySet())
System.out.println(e.getKey() + "" + e.getValue());
System.out.println("Insert map into another map");
Map<Integer, String> map2 = new HashMap<>();
map2.putAll(map);
map2.putAll(map1);
System.out.println(map2);
System.out.println();
System.out.println("Remove items from the map");
map.remove((3));
for (Map.Entry<Integer, String> e : map.entrySet())
System.out.println(e.getKey() + ""+ e.getValue());
System.out.println();
System.out.println();
System.out.println("Search specific key from the map");
```

```
System.out.println("Is the key '2' present? " +
 map.containsKey(2));
System.out.println("Is the key '6' present? " +
 map.containsKey(6));
System.out.println();
System.out.println("Get value of the specified key");
String val = (String)map.get(2);
System.out.println(val);
System.out.println();
}
}
```

Output :



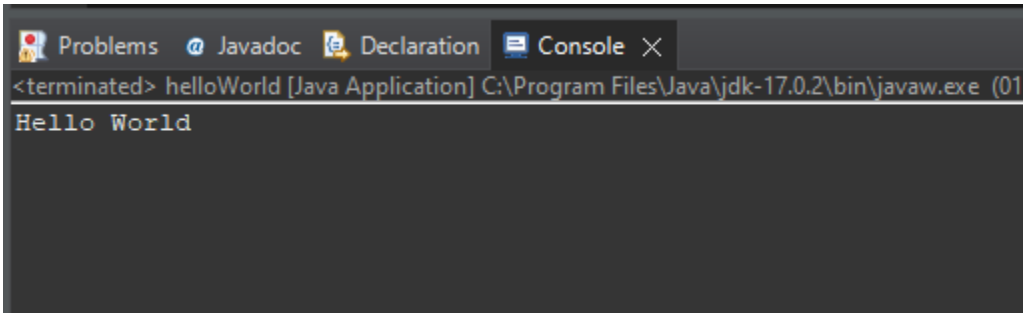
Lambda Expressions

Problem Statement 1 : Write a Java program using Lambda Expression to print “Hello World!”.

Solution :

```
package lambdaexpression;

interface HelloWorld1 {
	String sayHello(String name);
}
Public class helloWorld {
	public static void main(String args[]){
		HelloWorld1 helloWorld = (String name) -> { return "Hello " + name; };
		 System.out.println(helloWorld.sayHello("World"));
	}

}
```

Output :

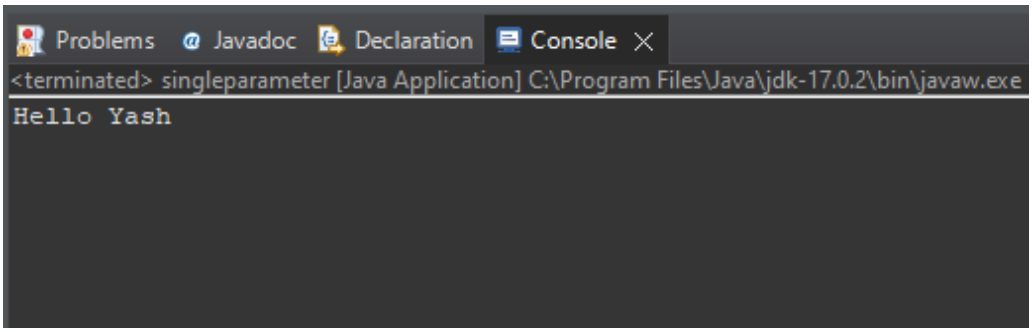


Problem Statement 2 : Write a Java program using Lambda Expression with single parameter.

Solution :

```
package lambdaexpression;

interface Say{

public String say(String name);

}

public class singleparameter{

public static void main(String[] args) {

            Say s1=(name)->{

return "Hello "+name;

        };

        System.out.println(s1.say("Yash"));

    }

}
```

Output :

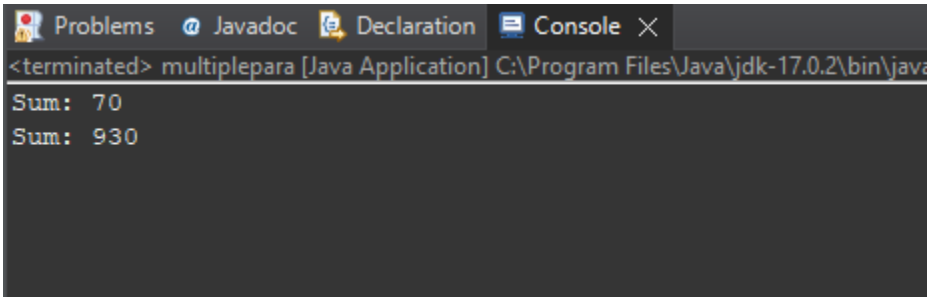


Problem Statement 3 : Write a Java program using Lambda Expression with multiple parameters to add two numbers.

Solution :

```
package lambdaexpression;
interface Add{
int add(int a,int b);
}

Public class multiplepara{
Public static void main(String[] args) {

      Add ad1=(a,b)->(a+b);
      System.out.println("Sum: " +ad1.add(50,20));

      Add ad2=(int a,int b)->(a+b);
      System.out.println("Sum: " +ad2.add(700,230));
   }

}
```

Output :



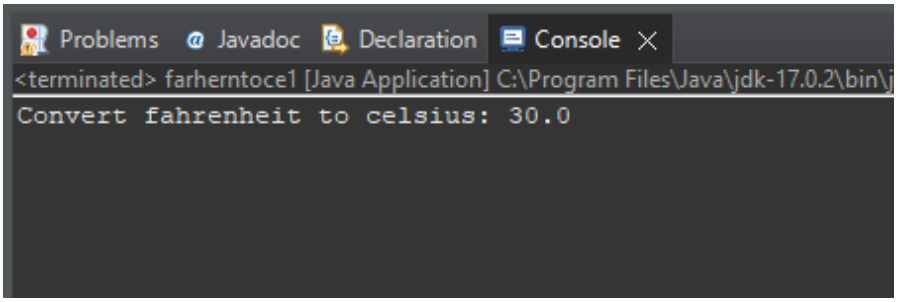
Problem Statement 4 : Write a Java program using Lambda Expression to calculate the following:

a. Convert Fahrenheit to Celsius

Solution :

```
package lambdaexpression;
interface temp
{
        Public double convert(double temp);

}
Public class farherntoce1 {
        Public staticvoid main(String[] args) {
                temp t1=(double a)->{
                        return((a-32)* 5/9);
                };
                System.out.print("Convert fahrenheit to celsius: "+ t1.convert(86));
        }


}
```

Output :

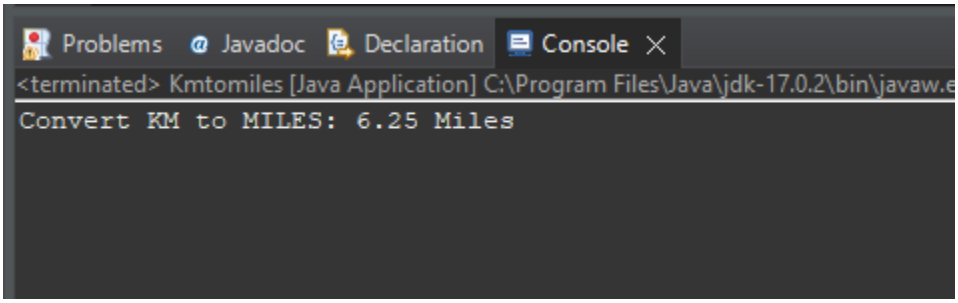


b. Convert Kilometers to Miles.

Solution :

```
package lambdaexpression;
interface temp1
{
	publicdouble convert(double temp);

}
publicclass Kmtomiles {
	publicstaticvoid main(String[] args) {
		temp t1=(double a)->{
			return(a/1.6);
		};
			System.out.print("Convert KM to MILES: "+ t1.convert(10)+ " Miles");
	}


}
```

Output :

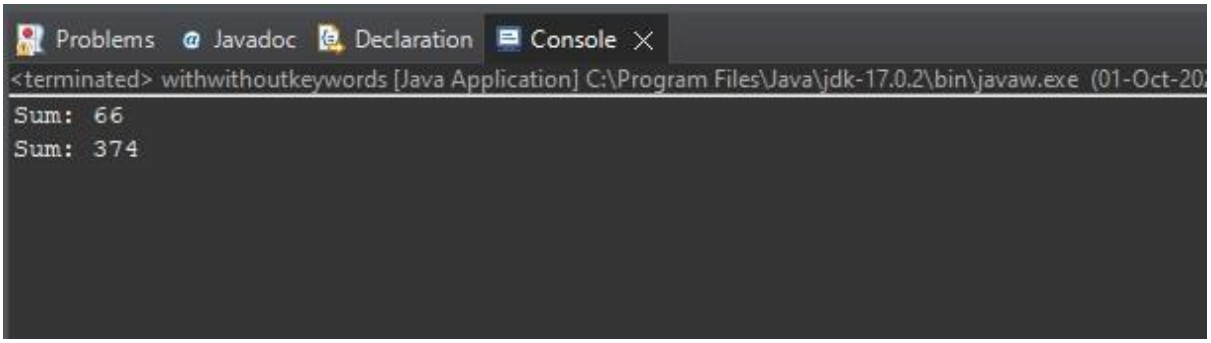


Problem Statement 5 : Write a Java program using Lambda Expression with or without return keyword.

Solution :

```
package lambdaexpression;
interface Add2{
int add(int a,int b);
}

Public class withwithoutkeywords {
Public static void main(String[] args) {

            // without return keyword
            Add2 ad1=(a,b)->(a+b);
        System.out.println("Sum: " +ad1.add(43,23));

        // with return keyword
        Add2 ad2=(int a,int b)->
        {
return (a+b);
        };
        System.out.println("Sum: " +ad2.add(54,320));
    }
}
```

Output :

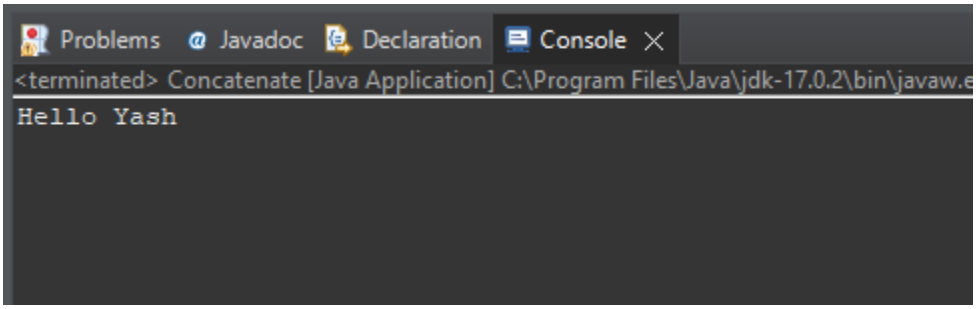


Problem Statement 6 : Write a Java program using Lambda Expression to concatenate two strings.

Solution :

```
package lambdaexpression;

interface conc1 {
	public String concat(String a,String b);
}
Public class concatenate {

	public static void main(String[] args) {
		conc1 s1 = (String a,String b)->{
			return (a+b);
		};
		System.out.println(s1.concat("Hello"," Vinit"));
	}
}
```

Output :



Web Application Development using JSP

Problem Statement 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.

Database table (phone1) :

```
CREATE TABLE phone1
(
id SERIAL PRIMARY KEY,
name varchar(50),
no varchar(50)
);
```

Index.jsp :

```
<%@page import="java.sql.*" %>
<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";

Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");

if(request.getParameter("delete")!=null)
{
int id=Integer.parseInt(request.getParameter("delete"));

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("delete from phone1 where id=? "); // delete query
pstmt.setInt(1,id);
pstmt.executeUpdate(); //execute query

con.close(); //close connection
}
}
catch(Exception e)
{
out.println(e);
}
```

```
%>
<html>

<head>

<title>JSP:Insert, Update, Delete </title>

</head>

<body>

<br>
<br>
<center>
<h1><a href="add.jsp">CLICK HERE TO ADD A NEW MOBILE NUMBER</a></h1>
</center>
<br>
<center>
<table border='1' cellpadding='23'>
</center>
<tr>
<th>ID</th>
<th>NAME</th>
<th>MOBILE NUMBER</th>
<th>UPDATE</th>
<th>DELETE</th>
</tr>
<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";

Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("select * from phone1"); //select query

ResultSet rs=pstmt.executeQuery(); //execute query and set in resultset object rs.

while(rs.next())
{
%>
<tr>
```

```
<td><%=rs.getInt(1)%></td>
<td><%=rs.getString(2)%></td>
<td><%=rs.getString(3)%></td>
<td><a href="update.jsp?edit=<%=rs.getInt(1)%>">Edit</a></td>
<td><a href="?delete=<%=rs.getInt(1)%>">Delete</a></td>

</tr>
<%
}

}
catch(Exception e)
{
out.println(e);
}

%>

</table>

</body>

</html>
```

Add.jsp :

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

<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";
Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");

if(request.getParameter("btn_add")!=null) //check button click event not null
{
String name,no;
name=request.getParameter("txt_name"); //txt_name
no=request.getParameter("txt_no"); //txt_owner

PreparedStatement pstmt=null; //create statement
```

```
pstmt=con.prepareStatement("insert into phone1(name,no)values(?,?)"); // insert query
pstmt.setString(1,name);
pstmt.setString(2,no);
pstmt.executeUpdate(); //execute query
con.close();  //close connection
out.println("Insert Successfully...! Click Home page.");// after insert record successfully message

}

}
catch(Exception e)
{
out.println(e);
}

%>

<html>

<head>

<title>JSP:Insert, Update, Delete using MySQL</title>

<!-- javascript for form validation-->
<script>

function validate()
{
var name = document.myform.txt_name;
var no = document.myform.txt_no;
if (name.value == "")
{
window.alert("please enter a name ?");
name.focus();
return false;
}
if (no.value == "")
{
window.alert("please enter a mobile number ?");
name.focus();
return false;
}

}

</script>
</head>
<body>
<form method="post" name="myform"  onsubmit="return validate();">
```

```
<center>
<h1>Insert Record</h1>
</center>
<table>
<tr>
<td><b>Name: </b></td>
<td><input type="text" name="txt_name"></td>
</tr>
<tr>
<td><b>Phone number:</b></td></b>
<td><input type="text" name="txt_no"></td>
</tr>
<tr>
<td><input type="submit" name="btn_add" value="Insert"></td>
</tr>
</table>
<center>
<h1><a href="index.jsp">Home page</a></h1>
</center>

</form>

</body>

</html>
```

Update.jsp :

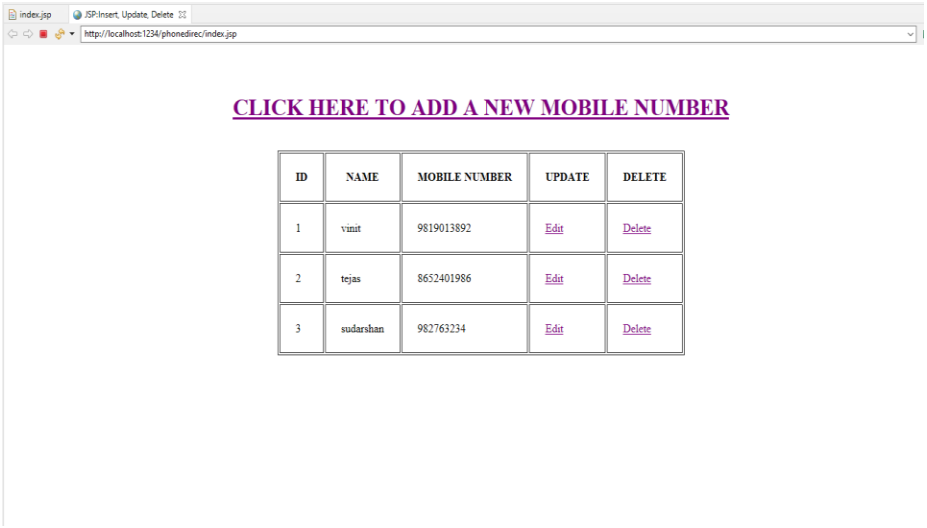
```
<%@ page import="java.sql.*" %>

<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";

Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");

if(request.getParameter("btn_update")!=null) //check button click event not null
{
int hide,name,no;;

String name_up,no_up;
```

```
hide=Integer.parseInt(request.getParameter("txt_hide")); //it is hidden id get for update record
name_up=request.getParameter("txt_name");
no_up=request.getParameter("txt_no"); //txt_name


PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("update phone1 set name=?,no=? where id=?"); // update query
pstmt.setString(1,name_up);
pstmt.setString(2,no_up);
pstmt.setInt(3,hide);
pstmt.executeUpdate(); //execute query

con.close(); //connection close

out.println("Update Successfully...! Click Back link."); //after update record successfully message
}

}
catch(Exception e)
{
out.println(e);
}

%>

<html>

<head>

<title>JSP:Insert, Update, Delete using MySQL</title>



<!-- javascript for form validation-->
<script>

function validate()
{
var name = document.myform.txt_name;
var no = document.myform.txt_no;

if (rno.value == "")
{
window.alert("please enter name ?");
name.focus();
return false;
}
if (name.value == "")
```

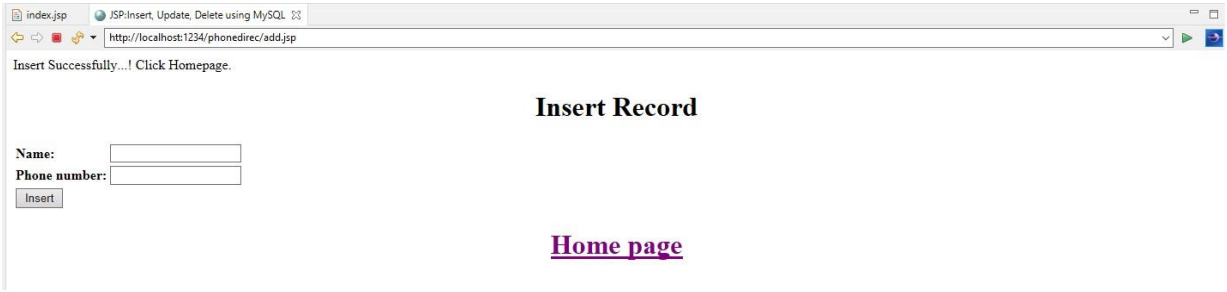
```
{
window.alert("please enter number ?");
name.focus();
return false;
}
}


</script>

</head>

<body>



<form method="post" name="myform"  onsubmit="return validate();">

<center>
<h1>Update Record</h1>
</center>

<table>
<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";

Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");

if(request.getParameter("edit")!=null)
{
int id=Integer.parseInt(request.getParameter("edit"));

String name,no;

PreparedStatement pstmt=null; // create statement

pstmt=con.prepareStatement("select * from phone1 where id=?"); // sql select query
pstmt.setInt(1,id);
ResultSet rs=pstmt.executeQuery(); // execute query store in resultset object rs.

while(rs.next())
```

```
{
id=rs.getInt(1);
name=rs.getString(2);
no=rs.getString(3);

%>
<tr>
<td>Name</td>
<td><input type="text" name="txt_name" value="<%=name%>"></td>
</tr>
<tr>
<td>Mobile Number</td>
<td><input type="text" name="txt_no" value="<%=no%>"></td>
</tr>
<tr>
<td><input type="submit" name="btn_update" value="Update"></td>
</tr>

<input type="hidden" name="txt_hide" value="<%=id%>">
<%
}
}
}
catch(Exception e)
{
out.println(e);
}
%>
</table>

<center>
<h1><a href="index.jsp">Back</a></h1>
</center>

</form>

</body>

</html>
```

Output:


index.jsp
JSP:Insert, Update, Delete
http://localhost:1234/phonedirec/index.jsp
CLICK HERE TO ADD A NEW MOBILE NUMBER
ID
NAME
MOBILE NUMBER
UPDATE
DELETE
1
vinit
9819013892
Edit
Delete
2
tejas
8652401986
Edit
Delete
3
sudarshan
982763234
Edit
Delete


index.jsp
JSP:Insert, Update, Delete using MySQL
http://localhost:1234/phonedirec/add.jsp
Insert Record
Name:
sushant
Phone number:
981234567
Insert
Home page


index.jsp
JSP:Insert, Update, Delete using MySQL
http://localhost:1234/phonedirec/add.jsp
Insert Successfully...! Click Homepage.
Insert Record
Name:
Phone number:
Insert
Home page

Data Output | Explain | Messages | Notifications

	id [PK] integer	name character varying (50)	no character varying (50)
1	1	vinit	9819013892
2	2	tejas	8652401986
3	3	sudarshan	98276234
4	4	sushant	981234567

Problem Statement 2. Write a JSP page to display the Registration form (Make your own assumptions)

Database table (studentreg1) :

```
CREATE TABLE studentreg1
(
id SERIAL PRIMARY KEY,
first_name varchar(50),
last_name varchar(50),
phn_number varchar(20),
address varchar(20),
course varchar(20),
college_name varchar(20)
);
```

Add.jsp :

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

<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";

Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");
```

```
if(request.getParameter("btn_add")!=null) //check button click event not null
{
String first_name,last_name,phn_number,address,course,college_name;
first_name=request.getParameter("txt_first_name"); //txt_name
last_name=request.getParameter("txt_last_name"); //txt_owner
phn_number=request.getParameter("txt_phn_number");
address=request.getParameter("txt_address");
course=request.getParameter("txt_course");
college_name=request.getParameter("txt_college_name");

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("insert into
studentreg1(first_name,last_name,phn_number,address,course,college_name)values(?,?,?,?,?,?)"); // insert query
pstmt.setString(1,first_name);
pstmt.setString(2,last_name);
pstmt.setString(3,phn_number);
pstmt.setString(4,address);
pstmt.setString(5,course);
pstmt.setString(6,college_name);
pstmt.executeUpdate(); //execute query
con.close();  //close connection
out.println("Insert Successfully...!");// after insert record successfully message

}

}
catch(Exception e)
{
out.println(e);
}

%>

<html>

<head>

<!-- javascript for form validation-->
<script>

function validate()
{
var first_name = document.myform.txt_first_name;
var last_name = document.myform.txt_last_name;
var phn_number = document.myform.txt_phn_number;
var address = document.myform.txt_address;
var course = document.myform.txt_course;
```

```
var college_name = document.myform.txt_college_name;
if (first_name.value == "")
{
window.alert("please enter a first name ?");
name.focus();
return false;
}
if (last_name.value == "")
{
window.alert("please enter a last name ?");
name.focus();
return false;
}
if (phn_number.value == "")
{
window.alert("please enter a mobile number ?");
name.focus();
return false;
}
if (address.value == "")
{
window.alert("please enter address ?");
name.focus();
return false;
}
if (course.value == "")
{
window.alert("please enter course ?");
name.focus();
return false;
}
if (college_name.value == "")
{
window.alert("please enter college name ?");
name.focus();
return false;
}

}

</script>

</head>

<body bgcolor="deea94">


<div align="center">
<form method="post" name="myform"  onsubmit="return validate();">
```

```
<center>
<h1><u>STUDENT REGISTRATION FORM</u></h1>
</center>

<br>
<table>
<tr>
<td><b>First Name: </b></td>
<td><input type="text" name="txt_first_name"></td>
</tr>

<tr>
<td><b>Last Name:</b></td></b>
<td><input type="text" name="txt_last_name"></td>
</tr>

<tr>
<td><b>Phone number:</b></td></b>
<td><input type="text" name="txt_phn_number"></td>
</tr>

<tr>
<td><b>Address:</b></td></b>
<td><input type="text" name="txt_address"></td>
</tr>
<tr>
<td><b>Course:</b></td></b>
<td><input type="text" name="txt_course"></td>
</tr>
<tr>
<td><b>College Name:</b></td></b>
<td><input type="text" name="txt_college_name"></td>
</tr>

<tr>
<td><br><input type="submit" name="btn_add" value="Submit"></br></td>
</tr>

</table>

<center>

<p><a><span>&#8595;</span><u>Click Below to list all the</u><span>&#8595;</span></a></p>
<p><a href="index.jsp">Registered Students Details</a></p>
</center>
</form>
</div>
</body>
```

```
</html>
```

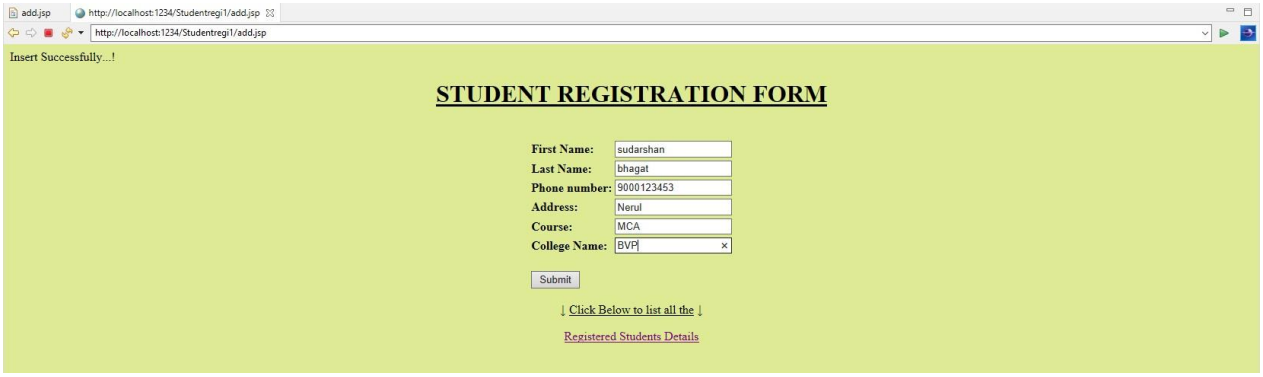
Index.jsp :

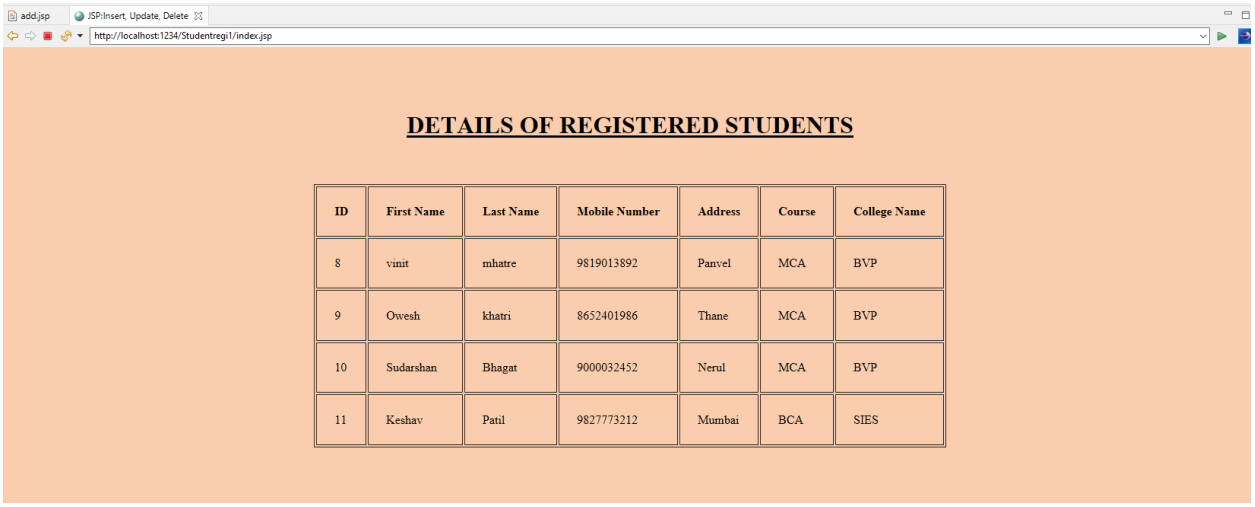
```
<%@page import="java.sql.*" %>
<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";

Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");

if(request.getParameter("delete")!=null)
{
int id=Integer.parseInt(request.getParameter("delete"));

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("delete from studentreg1 where id=? "); // delete query
pstmt.setInt(1,id);
pstmt.executeUpdate(); //execute query

con.close(); //close connection
}
}
catch(Exception e)
{
out.println(e);
}
%>
<html>

<head>
<title>JSP:Insert, Update, Delete </title>
</head>

<body bgcolor="F9CDAD">
<br>
<br>
<br>
<center>
<h1><u>DETAILS OF REGISTERED STUDENTS</u></h1>
</center>
```

```
<br><br>
<center>
<table border='1' cellpadding='23'>
</center>
<tr>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Mobile Number</th>
<th>Address</th>
<th>Course</th>
<th>College Name</th>
</tr>
<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";

Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);

PreparedStatement pstmt=null; //create statement

pstmt=con.prepareStatement("select * from studentreg1"); //select query

ResultSet rs=pstmt.executeQuery(); //execute query and set in resultset object rs.

while(rs.next())
{
%>
<tr>
<td><%=rs.getInt(1)%></td>
<td><%=rs.getString(2)%></td>
<td><%=rs.getString(3)%></td>
<td><%=rs.getString(4)%></td>
<td><%=rs.getString(5)%></td>
<td><%=rs.getString(6)%></td>
<td><%=rs.getString(7)%></td>
</tr>
<%
}

}
catch(Exception e)
{
```

```
out.println(e);
}
%>
</table>
</body>
</html>
```

OUTPUT :





ID	First Name	Last Name	Mobile Number	Address	Course	College Name
8	vinit	mhatre	9819013892	Panvel	MCA	BVP
9	Owesh	khatri	8652401986	Thane	MCA	BVP
10	Sudarshan	Bhagat	9000032452	Nerul	MCA	BVP
11	Keshav	Patil	9827773212	Mumbai	BCA	SIES

Data Output | Explain | Messages | Notifications

	id [PK] integer	first_name character varying (50)	last_name character varying (50)	phn_number character varying (20)	address character varying (20)	course character varying (20)	college_name character varying (20)
1	8	vinit	mhatre	9819013892	Panvel	MCA	BVP
2	9	Owesh	khatri	8652401986	Thane	MCA	BVP
3	10	Sudarshan	Bhagat	9000032452	Nerul	MCA	BVP
4	11	Keshav	Patil	9827773212	Mumbai	BCA	SIES

Problem Statement 3.Write a JSP program to add, delete and display the records from StudentMaster (RollNo, Name, Semester, Course) table.

Database table(student1):

```
CREATE TABLE student1
(
id SERIAL PRIMARY KEY,
rno varchar(50),
name varchar(50),
semester varchar(50),
course varchar(50)
);
```

Index.jsp :

```
<%@page import="java.sql.*" %>
<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";
Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");
if(request.getParameter("delete")!=null)
```

```
{
int id=Integer.parseInt(request.getParameter("delete"));
PreparedStatement pstmt=null; //create statement
pstmt=con.prepareStatement("delete from student1 where id=? "); // delete query
pstmt.setInt(1,id);
pstmt.executeUpdate(); //execute query
con.close(); //close connection
}
}
catch(Exception e)
{
out.println(e);
}
%>
<html>
<head>
<title>JSP:Insert, Update, Delete </title>
</head>
<body>
<center>
<h1><a href="add.jsp">Add Record</a></h1>
</center>

<table>
<tr>
<th>ID</th>
```

```
<th>Roll No</th>
<th>Name</th>
<th>Sem</th>
<th>Course</th>
<th>Update</th>
<th>Delete</th>
</tr>
<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";
Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
PreparedStatement pstmt=null; //create statement
pstmt=con.prepareStatement("select * from student1"); //select query
ResultSet rs=pstmt.executeQuery(); //execute query and set in resultset object rs.

while(rs.next())
{
%>
<tr>
<td><%=rs.getInt(1)%></td>
```

```
<td><%=rs.getString(2)%></td>

<td><%=rs.getString(3)%></td>

<td><%=rs.getString(4)%></td>

<td><%=rs.getString(5)%></td>

<td><a href="update.jsp?edit=<%=rs.getInt(1)%>">Edit</a></td>

<td><a href="?delete=<%=rs.getInt(1)%>">Delete</a></td>

</tr>

<%

}

}

catch(Exception e)

{

out.println(e);

}

%>

</table>

</body>

</html>
```

Add.jsp :

```
<%@ page import="java.sql.*" %>
<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";
Connection con =null;
Class.forName(driver).newInstance();
```

```
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");
if(request.getParameter("btn_add")!=null) //check button click event not null
{
String rno,name,semester,course;
rno=request.getParameter("txt_rno");
name=request.getParameter("txt_name"); //txt_name
semester=request.getParameter("txt_sem"); //txt_owner
course=request.getParameter("txt_course"); //txt_owner
PreparedStatement pstmt=null; //create statement
pstmt=con.prepareStatement("insert into student1(rno,name,semester,course)values(?,?,?,?)"); // insert query
pstmt.setString(1,rno);
pstmt.setString(2,name);
pstmt.setString(3,semester);
pstmt.setString(4,course);
pstmt.executeUpdate(); //execute query
con.close();  //close connection
out.println("Insert Successfully...! Click Back link.");// after insert record successfully message
}
}
catch(Exception e)
{
out.println(e);
}
%>
<html>
<head>
<title>JSP:Insert, Update, Delete using MySQL</title>


<!-- javascript for form validation-->
<script>
function validate()
{
var rno = document.myform.txt_rno;
var name = document.myform.txt_name;
var semester = document.myform.txt_sem;
var course = document.myform.txt_course;
if (rno.value == "")
{
window.alert("please enter rno ?");
name.focus();
return false;
}
if (name.value == "")
{
window.alert("please enter name ?");
name.focus();
return false;
```

```
}
if (semester.value == "")
{
window.alert("please enter sem ?");
owner.focus();
return false;
}
if (course.value == "")
{
window.alert("please enter course ?");
owner.focus();
return false;
}
}
</script>
</head>
<body>
<form method="post" name="myform"  onsubmit="return validate();">
<center>
<h1>Insert Record</h1>
</center>
<table>
<tr>
<td>Roll No</td>
<td><input type="text" name="txt_rno"></td>
</tr>
<tr>
<td>Name</td>
<td><input type="text" name="txt_name"></td>
</tr>
<tr>
<td>Sem</td>
<td><input type="text" name="txt_sem"></td>
</tr>
<tr>
<td>Course</td>
<td><input type="text" name="txt_course"></td>
</tr>
<tr>
<td><input type="submit" name="btn_add" value="Insert"></td>
</tr>
</table>
<center>
<h1><a href="index.jsp">Back</a></h1>
</center>
</form>
</body>
</html>
```

Update.jsp :

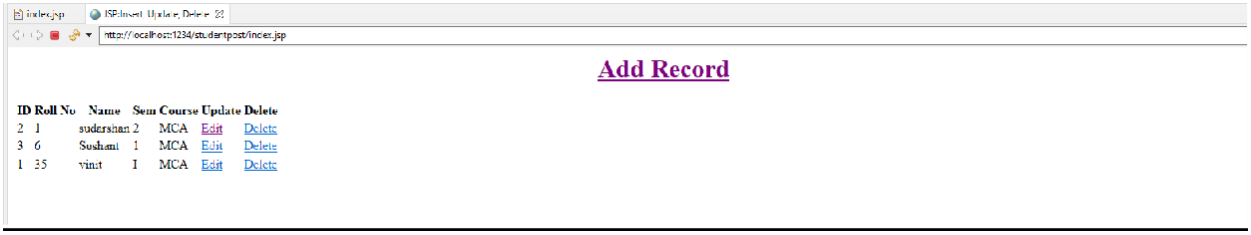
```
<%@ page import="java.sql.*" %>

<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";
Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");
if(request.getParameter("btn_update")!=null) //check button click event not null
{
int hide,rno,name,semester,course;;
String rno_up,name_up,semester_up,course_up;
hide=Integer.parseInt(request.getParameter("txt_hide")); //it is hidden id get for update record
rno_up=request.getParameter("txt_rno");
name_up=request.getParameter("txt_name"); //txt_name
semester_up=request.getParameter("txt_semester");
course_up=request.getParameter("txt_course");
PreparedStatement pstmt=null; //create statement
pstmt=con.prepareStatement("update student1 set rno=?,name=?, semester=?, course=? where id=?"); // update query
pstmt.setString(1,rno_up);
pstmt.setString(2,name_up);
pstmt.setString(3,semester_up);
pstmt.setString(4,course_up);
pstmt.setInt(5,hide);
pstmt.executeUpdate(); //execute query
con.close(); //connection close
out.println("Update Successfully...! Click Back link."); //after update record successfully message
}
}
catch(Exception e)
{
out.println(e);
}
%>
<html>
<head>
<title>JSP:Insert, Update, Delete using MySQL</title>

<!-- javascript for form validation-->
<script>

function validate()
{
```

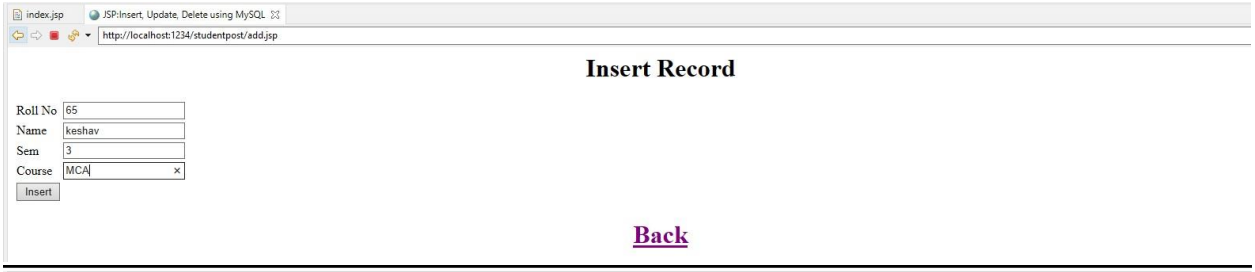
```
var rno = document.myform.txt_rno;
var name = document.myform.txt_name;
var semester = document.myform.txt_semester;
var course = document.myform.txt_course;
if (rno.value == "")
{
window.alert("please enter rno ?");
name.focus();
return false;
}
if (name.value == "")
{
window.alert("please enter name ?");
name.focus();
return false;
}
if (semester.value == "")
{
window.alert("please enter sem ?");
owner.focus();
return false;
}
if (course.value == "")
{
window.alert("please enter course ?");
owner.focus();
return false;
}}
</script>
</head>
<body>

<form method="post" name="myform"  onsubmit="return validate();">
<center>
<h1>Update Record</h1>
</center>
<table>
<%
try
{
String driver ="org.postgresql.Driver";
String url ="jdbc:postgresql://localhost:5432/postgres";
String username ="postgres";
String password ="admin";
Connection con =null;
Class.forName(driver).newInstance();
con = DriverManager.getConnection(url,username,password);
System.out.println("Opened database successfully");
if(request.getParameter("edit")!=null)
```

```
{
int id=Integer.parseInt(request.getParameter("edit"));
String rno,name,semester,course;
PreparedStatement pstmt=null; // create statement
pstmt=con.prepareStatement("select * from student1 where id=?"); // sql select query
pstmt.setInt(1,id);
ResultSet rs=pstmt.executeQuery(); // execute query store in resultset object rs.
while(rs.next())
{
id=rs.getInt(1);
rno=rs.getString(2);
name=rs.getString(3);
semester=rs.getString(4);
course=rs.getString(5);
%>
<tr>
<td>Roll NO</td>
<td><input type="text" name="txt_rno" value="<%=rno%>"></td>
</tr>
<tr>
<td>Name</td>
<td><input type="text" name="txt_name" value="<%=name%>"></td>
</tr>
<tr>
<td>Sem</td>
<td><input type="text" name="txt_semester" value="<%=semester%>"></td>
</tr>
<tr>
<td>Course</td>
<td><input type="text" name="txt_course" value="<%=course%>"></td>
</tr>
<tr>
<td><input type="submit" name="btn_update" value="Update"></td>
</tr>
<input type="hidden" name="txt_hide" value="<%=id%>">
<%
}
}
}
catch(Exception e)
{
out.println(e);
}
%>
</table>
<center>
<h1><a href="index.jsp">Back</a></h1>
</center>
</form>
```

</body></html>

Output:





Data Output | Explain | Messages | Notifications

	id [PK] integer	rno character varying (50)	name character varying (50)	semester character varying (50)	course character varying (50)
1	2	1	sudarshan	2	MCA
2	3	6	Sushant	1	MCA
3	1	35	vinit	I	MCA
4	4	65	keshav	3	MCA

Problem Statement 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:
a. 1 to 7 year at 5.35%
b. 8 to 15 year at 5.5%
c. 16 to 30 year at 5.75%

Cal.jsp :

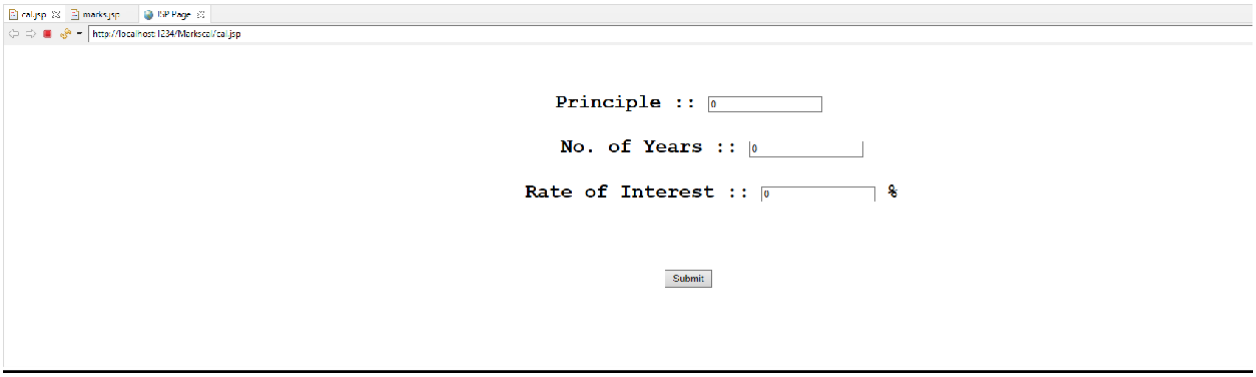
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body><br><br><center>
<form action="test.jsp "><pre>
<h1>Principle :: <input type=text name=principle value=0 "><br>
   No. of Years :: <input type=text name=year value=0 "><br>
   Rate of Interest :: <input type=text name=interest value=0 "> %<br>
<br>
<input type=submit value="Submit"></h1>

</pre></form></center>
</body>
</html>
```

Test.jsp:

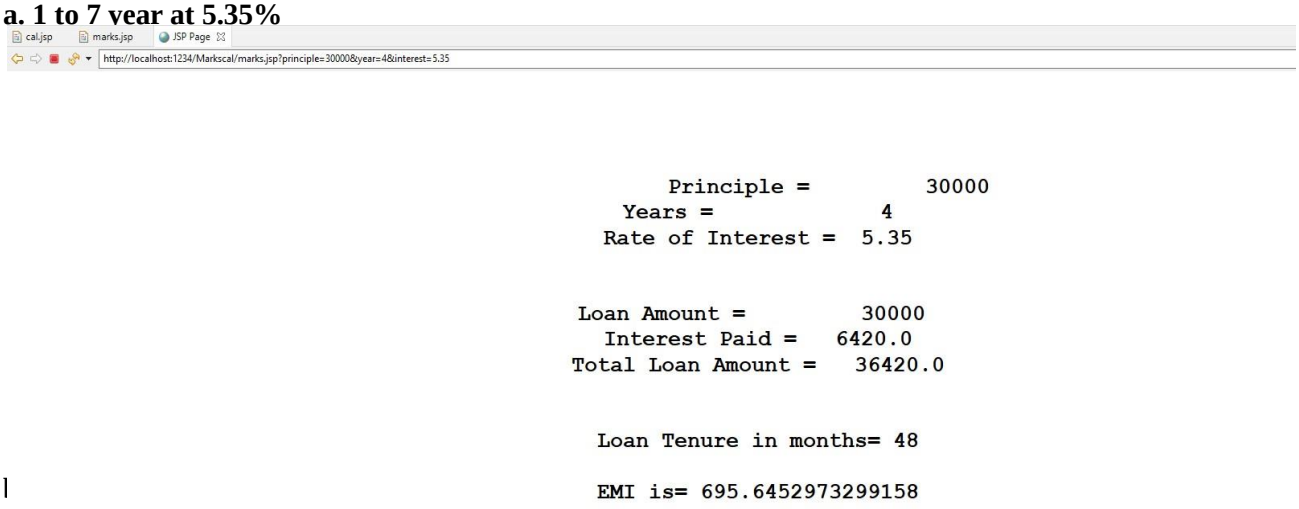
```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body><br><br><center><pre><H1>
<%

String ns= request.getParameter("principle");
String ns1= request.getParameter("year");
```

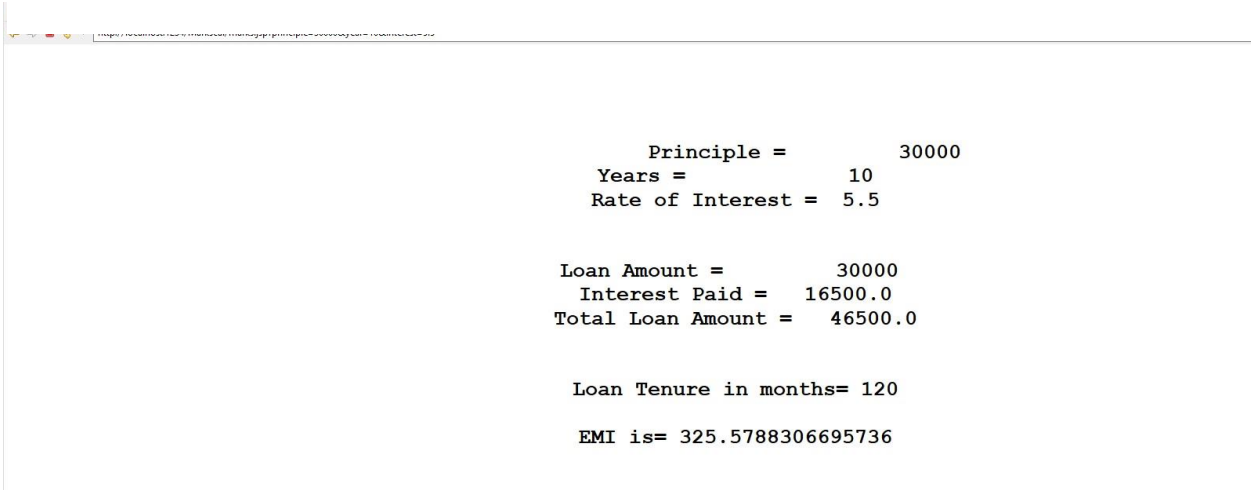
```
String ns2= request.getParameter("interest");
int n1=Integer.parseInt(ns);
int n2=Integer.parseInt(ns1);
float n3 = Float.valueOf(ns2);

double si=((n1*n2*n3)/100);
double x;
x=n1+si;

double r = (n3)/(12*100);

int mon;
mon=((n2)*12);
double emi= (n1*r*Math.pow(1+r,mon))/(Math.pow(1+r,mon)-1);




%>
<%
out.println("Principle =         "+n1);
out.println(" Years =            "+n2);
out.println(" Rate of Interest = "+n3);
out.println("<br>");
out.println("Loan Amount =         "+n1);
out.println(" Interest Paid = "+si);
out.println(" Total Loan Amount =  "+x);
out.println("<br>");
out.print(" Loan Tenure in months= " +mon);
out.println("<br>");
out.print(" EMI is= "+emi+"\n");

%>
</H1>
</pre>
</center>
</body></body>
</html>
```

Output:



a. 1 to 7 year at 5.35%





c. 16 to 30 year at 5.75%

Principle = 30000
Years = 15
Rate of Interest = 5.75

Loan Amount = 30000
Interest Paid = 25875.0
Total Loan Amount = 55875.0

Loan Tenure in months= 180

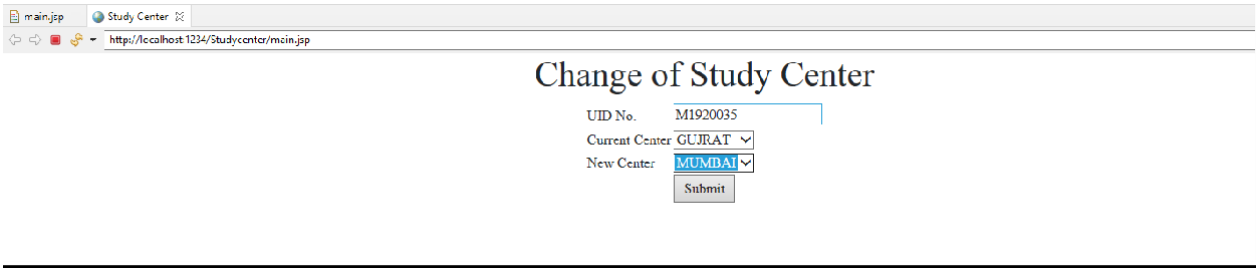
EMI is= 249.12302778152468

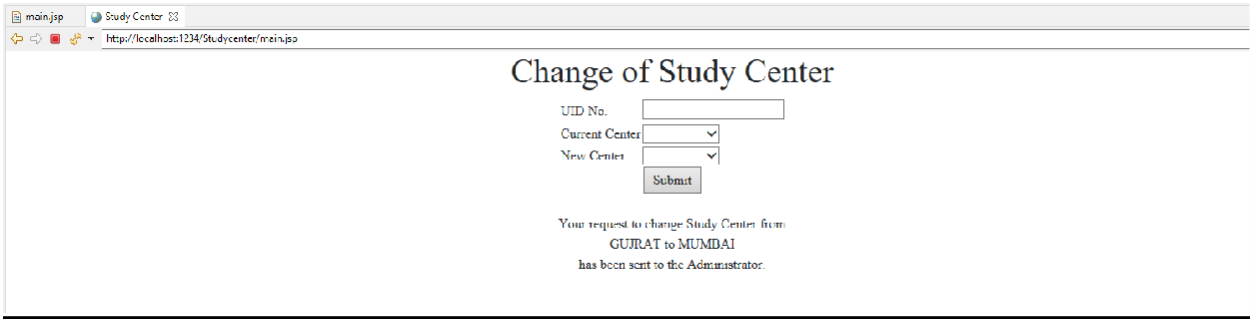
Problem Statement 5. Write a program using JSP that displays a webpage consisting Application form for change of Study Center which can be filled by any student who wants to change his/ her study center. Make necessary assumptions

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Study Center</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.4.0/font/bootstrap-icons.css">
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet"
integrity="sha384-BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqIj61tLrc4wSX0szH/Ev+nYRRuWlolflfl"
crossorigin="anonymous">

</head>
<body>
<center>

<h1>Change of Study Center</h1>

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

<table>
<tr>
<td>UID No.</td>
<td><input type="text" name="uid" required/></td>
</tr>
<tr>
```

```
<td>
Current Center
</td>
<td>
<select name="currentCenter" required>
<option selected disabled hidden></option>
<option value="MUMBAI">MUMBAI</option>
<option value="PUNE">PUNE</option>
<option value="GUJRAT">GUJRAT</option>
</select>
</td>
</tr>
<tr>
<td>
New Center
</td>
<td>
<select name="newCenter" required>
<option selected disabled hidden></option>
<option value="MUMBAI">MUMBAI</option>
<option value="PUNE">PUNE</option>
<option value="GUJRAT">GUJRAT</option>
</select>
</td>
</tr>

</table>
<input type="submit"  value="Submit"/>
</form>

</center>

<%

if(request.getParameter("uid") != null&& request.getParameter("currentCenter") != null&&
request.getParameter("newCenter") != null){
out.println("<center><br>Your request to change Study Center from <br>" +
request.getParameter("currentCenter") + " to " + request.getParameter("newCenter") + "<br> has been sent to the
Administrator.</center>");
}

%>

</body>
</html>
```

Output:





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

Main.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>JSP EXAMPLE</title>
</head>
<body>
<%@ include file = "header.jsp" %>
<center>
<%! int data=50; %>
<%= "Value of the variable is:"+data %>
<%!
double circle(int n){ return 3.14*n*n;}
%></br>
<%= "Area of circle is:"+ circle(3) %></br>
<%!
int rectangle(int l,int b){ return l*b;}
%>
<%= "Area of rectangle is:"+rectangle(3,4
) %></br>
```

```
<%!
int perimeter(int x,int y){
 int peri=2*(x+y);
return peri;}
%>
<%= "Perimeter of rectanlge:"+perimeter(5,6
) %></br>
<p>Thanks for visiting my page.</p>
</center>
<%@ include file = "footer.jsp" %>
</body>
</html>
```

Header.jsp :

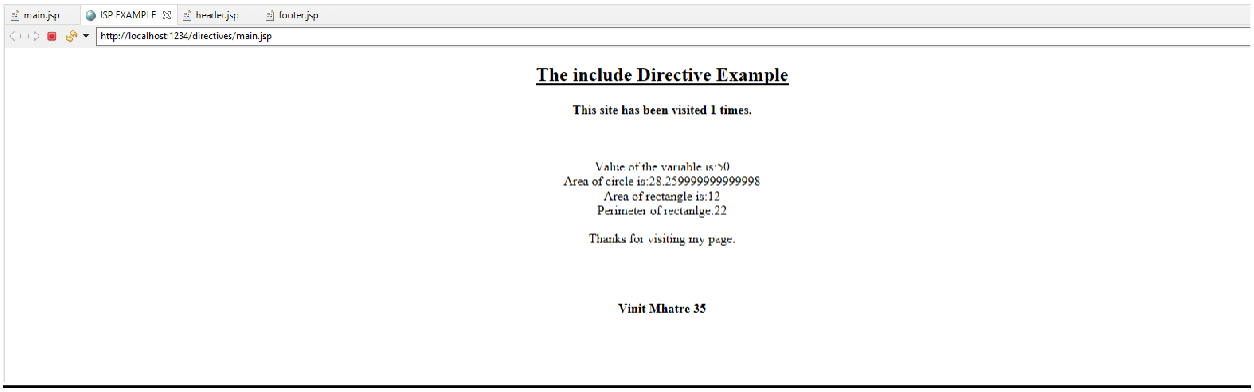
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>

<%!
 int pageCount = 0;
 void addCount() {
 pageCount++;
 }
%>
<% addCount(); %>
<html>
<head>
<title>JSP declaration, scriptlet, directives, expression, header and footer Example</title>
</head>
<body>
<center>
<h2><u>The include Directive Example</u></h2>
<p><b>This site has been visited <%= pageCount %> times.</b></p>
</center>
<br/><br/>

</body>
</html>
```

Footer.jsp:

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

```
</head>
<body>
<br/><br/>
<center><p><b>Vinit Mhatre 35</b></p></center></body></html>
```

Output:



Problem Statement 7. Write a JSP program that demonstrates the use of session or cookies.

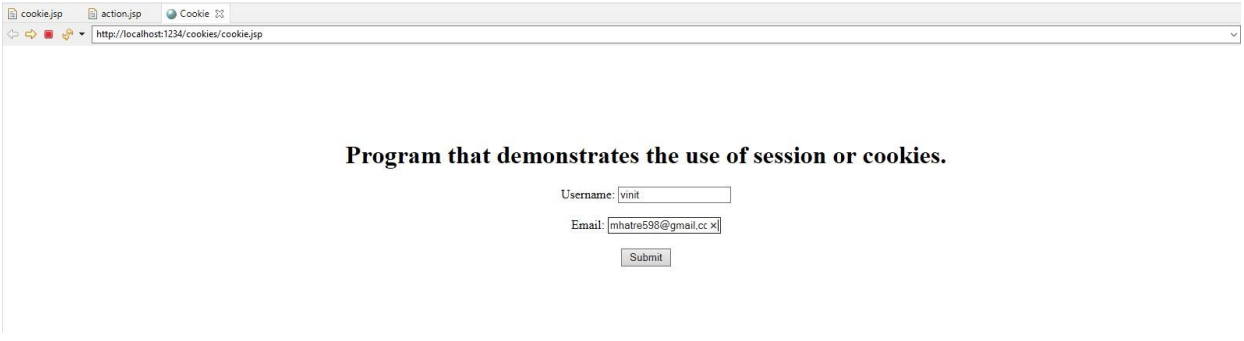
Cookie.jsp :

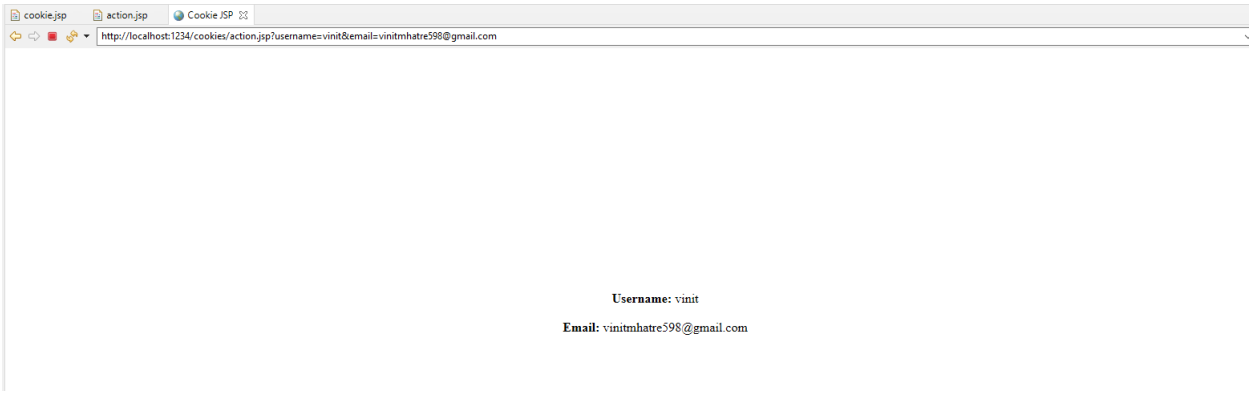
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Cookie</title>
</head>
<body><center>
<br><br><br><br><br>
<form action="action.jsp" method="GET">
<h1>Program that demonstrates the use of session or cookies.</h1>
Username: <input type="text" name="username">
<br><br>
Email: <input type="text" name="email" />
<br><br>
<input type="submit" value="Submit" />
</center>
</form>
</body>
</html>
```

Action.jsp :

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN""http://www.w3.org/TR/html4/loose.dtd">
<%

Cookie username = new Cookie("username",
request.getParameter("username"));
Cookie email = new Cookie("email",
request.getParameter("email"));

username.setMaxAge(60*60*10);
email.setMaxAge(60*60*10);

// Add both the cookies in the response header.
response.addCookie( username );
response.addCookie( email );
%>


<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Cookie JSP</title>
</head>
<body>
<br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
<center>
<b>Username:</b>
<%= request.getParameter("username")%><br><br>
<b>Email:</b>
<%= request.getParameter("email")%>
</center>
</body>
</html>
```

Output:




cookie.jsp
action.jsp
Cookie JSP
http://localhost:1234/cookies/action.jsp?username=vinit&email=vinitmhatre598@gmail.com
Username: vinit
Email: vinitmhatre598@gmail.com

Spring Framework

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

Solution :

HelloWorld.java

```
package spring1;

publicclass HelloWorld {

	String name;

	public String getName() {
		return name;
	}

	publicvoid setName(String name) {
		this.name = name;
	}

	@Override
	public String toString() {
		return "Hello World, I'm " + name + ".";
	}
}
```

appctx3.xml

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

	<bean id="hw" class="spring1.HelloWorld">
		<property name="name"value="Yash"/>
	</bean>

</beans>
```

TestHelloWorld.java

```
package spring1;

import org.springframework.context.support.ClassPathXmlApplicationContext;

publicclass TestHelloWorld {

	publicstaticvoid main(String[] args) {

		ClassPathXmlApplicationContext app = new ClassPathXmlApplicationContext("appctx3.xml");
		HelloWorld hw = (HelloWorld) app.getBean("hw");

		System.out.println(hw.toString());

	}

}
```

Output :

```
<terminated> TestHelloWorld (3) [Java Application] C:\Program Files\Java\jre1.8.0_202\bin\javaw.exe (15-Dec-2023, 11:33:10 am)
Hello World, I'm Yash.
```

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

Solution:

Account.java

```
package spring1;

publicclass Account {

	int id;
	String name;
	int balance;

	public Account(int id, String name, int balance) {
		super();
		this.id = id;
		this.name = name;
		this.balance = balance;
	}
	publicint getId() {
		return id;
	}
	publicvoid setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	publicvoid setName(String name) {
		this.name = name;
	}
	publicint getBalance() {
		return balance;
	}
	publicvoid setBalance(int balance) {
		this.balance = balance;
	}
	@Override
	public String toString() {
		return "Account [id=" + id + ", name=" + name + ", balance=" + balance + "]";
	}

}
```

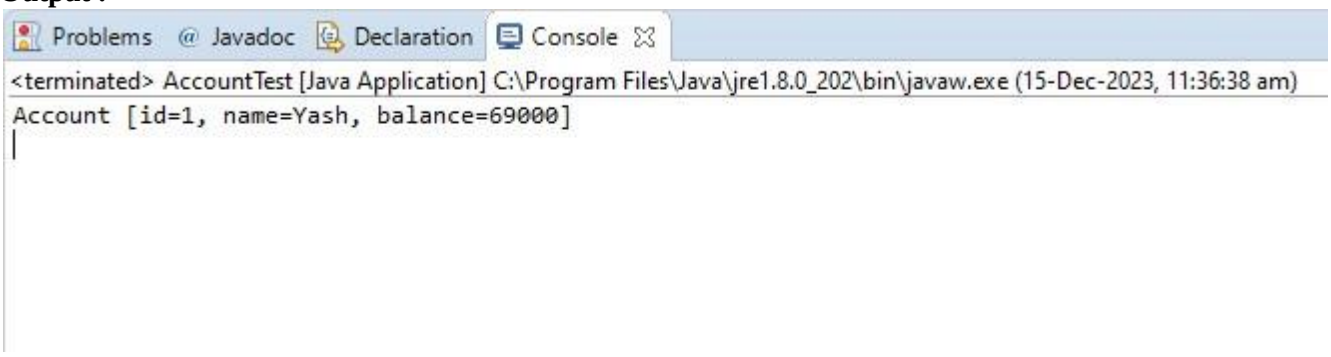
appctx2.xml

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

<bean id="Account" class="spring1.Account">

	<constructor-arg name="id" value="1"></constructor-arg>
	<constructor-arg name="name" value="vinit"></constructor-arg>
	<constructor-arg name="balance" value="69000"></constructor-arg>

</bean>

</beans>
```

AccountTest.java

```
package spring1;

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

public class Accounttest {

	public static void main(String[] args) {

		ApplicationContext con = new ClassPathXmlApplicationContext("appctx2.xml");
		Account acc = (Account) con.getBean("Account");

		System.out.println(acc.toString());

	}
}
```

Output :



Problem Statement 3 : Write a program to demonstrate dependency injection via Constructor.

Solution:

Singer.java

```
package spring1;

publicclass Singer {
	String name;
	int age;
	public String getName() {
		return name;
	}
	publicvoid setName(String name) {
		this.name = name;
	}
	publicint getAge() {
		return age;
	}
	publicvoid setAge(int age) {
		this.age = age;
	}
void displayInfo()
{
	System.out.println("Name:" +name+" Age:" +age);
}
}
```

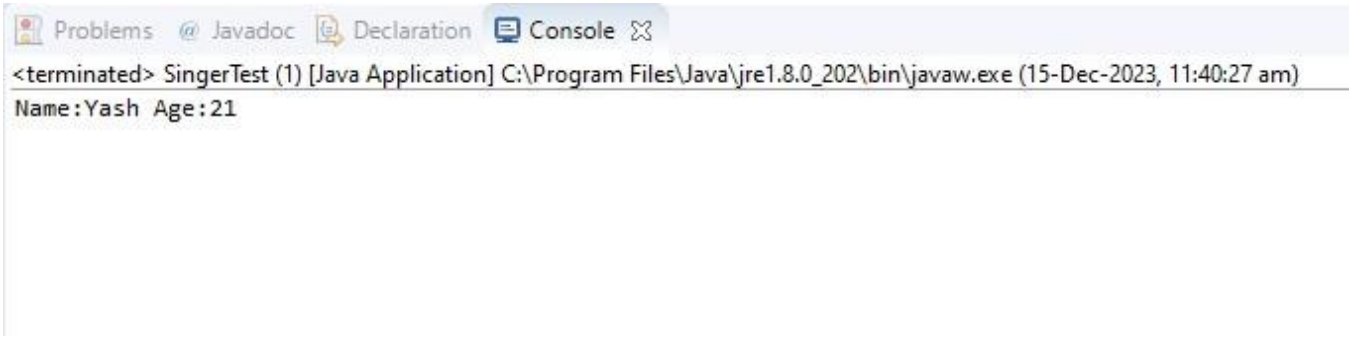
appctx.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
<bean id="Singer" class="spring1.Singer">
<property name="name" value="vinit"></property>
<property name="age" value="21"></property>
</bean>

</beans>
```

SingerTest.java

```
package spring1;

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

public class SingerTest {

	private static ApplicationContext ctx;
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		ctx=new ClassPathXmlApplicationContext("appctx.xml");
		Singer singer=(Singer)ctx.getBean("Singer");
			singer.displayInfo();



	}

}
```

Output :



Aspect Oriented Programming

Problem Statement 1 : Write a program to demonstrate Spring AOP – before advice.

Solution :

beforeaop.java

```
package bvimit.edu;

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

@Aspect
public class beforeaop {

	@Pointcut("execution(int beforeoperation.*(..))")
	public void p(){}

	@Before("p()")
	public void myadvice(JoinPoint jp)
	{
		System.out.println("before advice");
	}
}
```

beforeoperation.java

```
package bvimit.edu;

publicclass beforeoperation {
publicvoid msg() {System.out.println("method 1");}
publicint m(){System.out.println("method 2 with return");return 2;}
publicint k(){System.out.println("method 3 with return");return 3;}
    }
```

aopctx1.xml

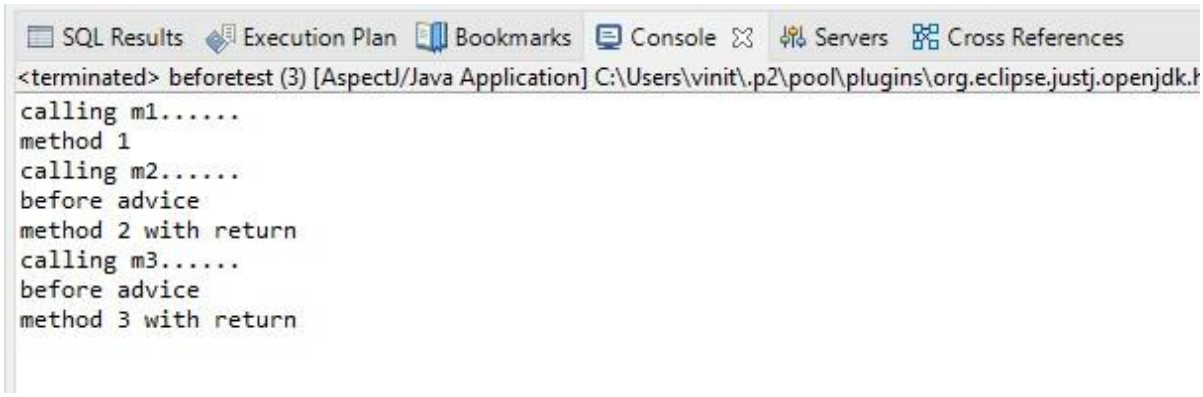
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="opBean" class="bvimit.edu.beforeoperation"></bean>

<bean id="trackMyBean" class="bvimit.edu.beforeaop"></bean>



<bean class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean>
```

</beans>

beforetest.java

```
package bvimit.edu;

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

public class beforetest {

	public static void main(String[] args) {
		ApplicationContext context = new ClassPathXmlApplicationContext("aopctx1.xml");
		beforeoperation e = (beforeoperation) context.getBean("opBean");
		System.out.println("calling m1. .... ");
		e.msg();
		System.out.println("calling m2. .... ");
		e.m();
		System.out.println("calling m3. .... ");
		e.k();


	}

}
```

Output :



Problem Statement 2 : Write a program to demonstrate Spring AOP – after advice.

Solution :

Afteraopdata.java

```
package bvimit.edu;

import  org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class afteraopdata {

	@Pointcut("execution(int afteroperation.*(..))")
	public void p(){}

	@After("p()")
	public void myadvice(JoinPoint jp)
	{
		System.out.println("after advice");
	}
}
```

afteroperation.java

```
package bvimit.edu;

publicclass afteroperation {
publicvoid msg() {System.out.println("method 1");}
publicint m(){System.out.println("method 2 with return");return 2;}
publicint k(){System.out.println("method 3 with return");return 3;}
    }
```

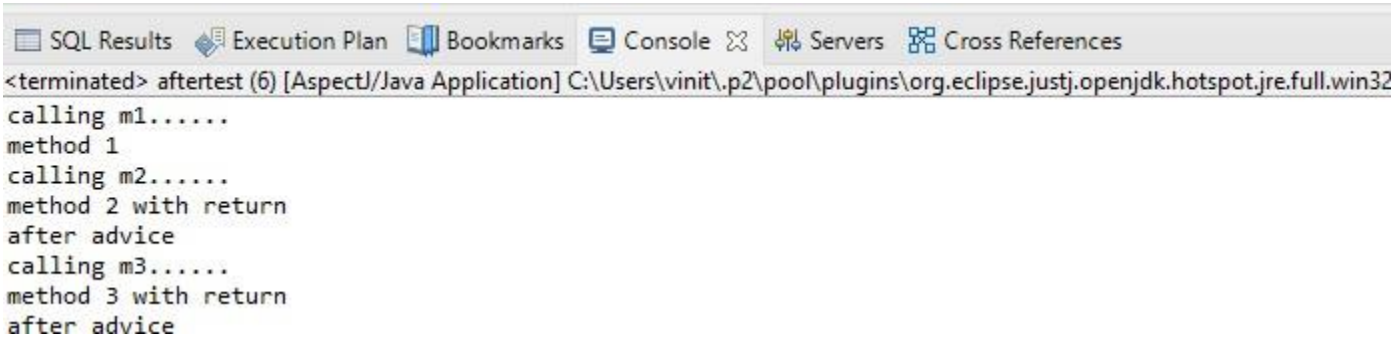
aopctx.xml

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

<bean id="opBean" class="bvimit.edu.afteroperation"></bean>

<bean id="trackMyBean" class="bvimit.edu.afteraopdata"></bean>



<bean class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean>
</beans>
```

aftertest.java

```
package bvimit.edu;

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

public class aftertest {

	public static void main(String[] args) {
		ApplicationContext context = new ClassPathXmlApplicationContext("aopctx.xml");
		afteroperation e = (afteroperation) context.getBean("opBean");
		System.out.println("calling m1. .... ");
		e.msg();
		System.out.println("calling m2. .... ");
		e.m();
		System.out.println("calling m3. .... ");
		e.k();


	}

}
```

Output :



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

Solution :

Bankaopdata.java

```
package bvimit.edu;

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class Bankaopdata {

	@Pointcut("execution(* Bank.*(..))")
	public void a() {}

	@Around("a()")
	public Object myadvice(ProceedingJoinPoint p)throws Throwable
	{
		System.out.println("Around concern Before calling actual method");
		Object obj=p.proceed();
		System.out.println("Around Concern After calling actual method");
		return obj;
	}
}
```

Bank.java

```
package bvimit.edu;

publicclass Bank {
	publicvoid welcome() {System.out.println("welcome to bank");}
	publicint icici() {System.out.println("icici bank interest rate");return 7;}
	publicint pnb() {System.out.println("pnb bank interest rate");return 6;}

}
```

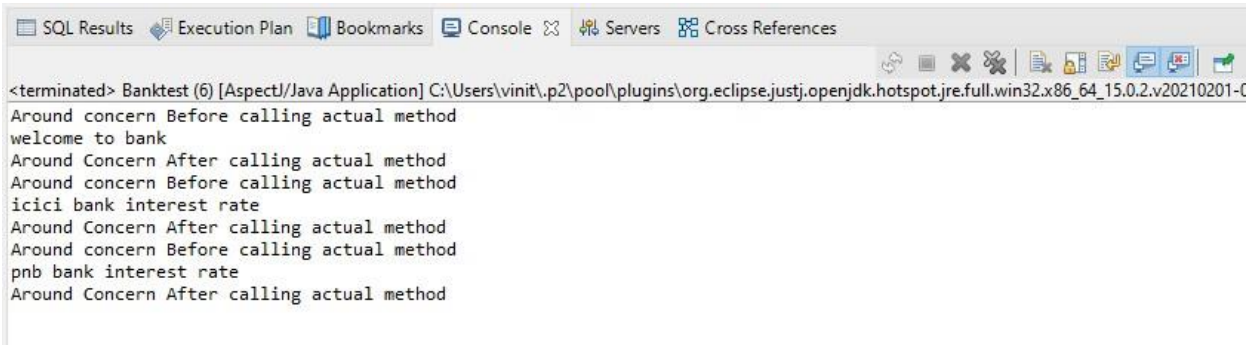
Bankaopdata.xml

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

<bean id="opBean" class="bvimit.edu.Bank"></bean>
<bean id="trackMyBean" class="bvimit.edu.Bankaopdata"></bean>

<bean class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean>
</beans>
```

Banktest.java

```
package bvimit.edu;

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

publicclass Banktest {

	privatestatic ApplicationContext context;

	publicstaticvoid main(String[] args) {
		context = new ClassPathXmlApplicationContext("Bankaopdata.xml");

		Bank e =(Bank) context.getBean("opBean");
		System.out.println("Calling welcome method...");
		e.welcome();
		System.out.println("Calling icici method...");
		e.icici();
		System.out.println("Calling pnb method...");
		e.pnb();
	}

}
```

Output :



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

Solution :

Bankaopdata.java

```
package bvimit.edu;

import org.aspectj.lang.JoinPoint;
importorg.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.AfterReturning;
importorg.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
importorg.aspectj.lang.annotation.Pointcut;

@Aspect
publicclass Bankaopdata {

	@AfterReturning(
			pointcut ="execution(* Bank.*(..))",
			returning="result")
publicvoid myadvice(JoinPoint jp,Object result)
{
	System.out.println("AfterReturning concern");
	System.out.println("Result in advice" +result);
	}
}
```

Bank.java

```
package bvimit.edu;

publicclass Bank {
	publicvoid welcome() {System.out.println("welcome to bank");}
	publicint icici() {System.out.println("icici bank interest rate");return 7;}
	publicint pnb() {System.out.println("pnb bank interest rate");return 6;}

}
```

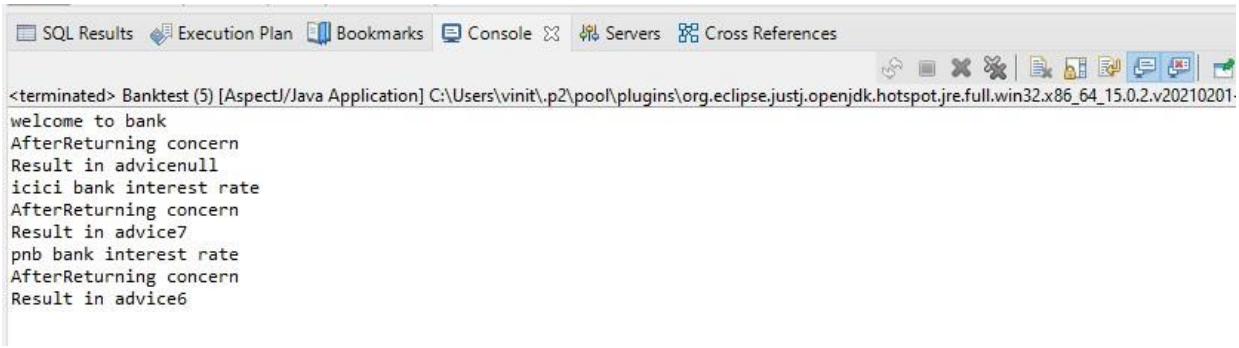
Bankaopdata.xml

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

<bean id="opBean" class="bvimit.edu.Bank"></bean>
<bean id="trackMyBean" class="bvimit.edu.Bankaopdata"></bean>

<bean class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean>
</beans>
```

Banktest.java

```
package bvimit.edu;

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

publicclass Banktest {

	privatestatic ApplicationContext context;

	publicstaticvoid main(String[] args) {
		context = new ClassPathXmlApplicationContext("Bankaopdata.xml");

		Bank e =(Bank) context.getBean("opBean");
		//System.out.println("Calling welcome method...");
		e.welcome();
		//System.out.println("Calling icici method...");
		e.icici();
		//System.out.println("Calling pnb method...");
		e.pnb();
	}

}
```

Output :



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

Solution :

Operationaop_at.java

```
package bvimit.edu;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Aspect;

@Aspect
publicclass Operationaop_at {
@AfterThrowing(
                        pointcut = "execution(* Operation_at.*(..))", throwing = "error")
        publicvoid myadvice(JoinPoint jp, Throwable error)
        {
                System.out.println("AfterThrowing concern");
                System.out.println("Exception is: "+error);
                System.out.println("end of after throwing advice ... ");
        }
        }
```

Operation_at.java

```
package bvimit.edu;
publicclass Operation_at {

        publicvoid validate(int att)throws Exception{
                if(att<75) {
                        thrownew ArithmeticException("Not eligible for exam");
                }
                else {
                        System.out.println("Eligible for exam");
                }
        }
}
```

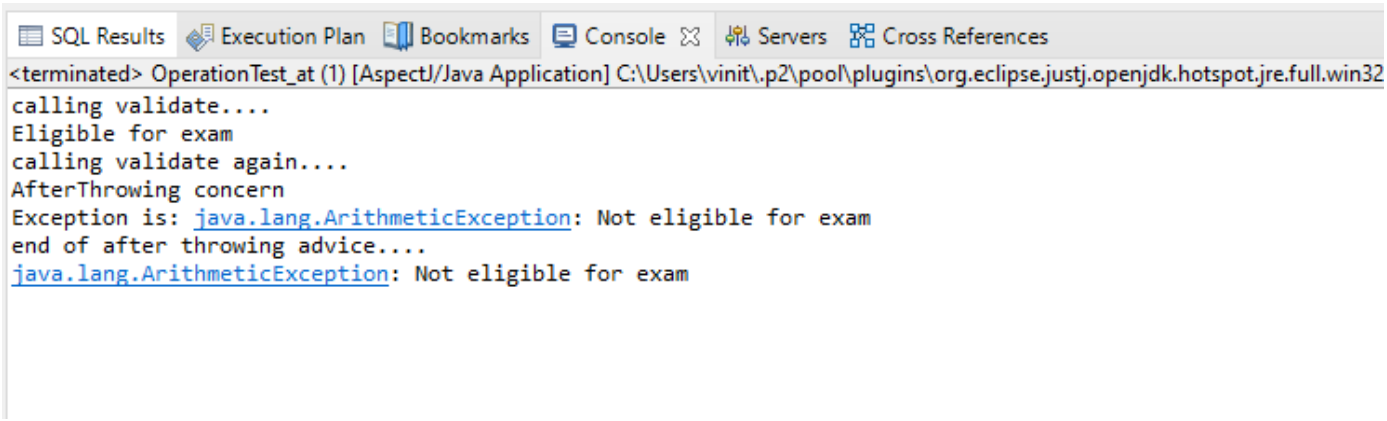
validctx.xml

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


<bean id="opBean" class="bvimit.edu.Operation_at"></bean>

<bean id="trackMyBean" class="bvimit.edu.Operationaop_at"></bean>

<bean
class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean></beans>
```

TestValidation.java

```
package bvimit.edu;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class OperationTest_at {
private static ApplicationContext context;
                public static void main(String[] args) {
ApplicationContext context = new ClassPathXmlApplicationContext("validctx.xml");
                        Operation_at op = (Operation_at) context.getBean("opBean");
                        System.out.println("calling validate....");
                         try {
                                op.validate(85);
                         }catch(Exception e){System.out.println(e);}

                         System.out.println("calling validate again... ");

                         try {
                                op.validate(25);
                         }catch(Exception e){System.out.println(e);}
                         }
                        }
```

Output :


SQL Results
Execution Plan
Bookmarks
Console
Servers
Cross References
<terminated> OperationTest_at (1) [AspectJ/Java Application] C:\Users\vinit\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
calling validate....
Eligible for exam
calling validate again....
AfterThrowing concern
Exception is: java.lang.ArithmeticException: Not eligible for exam
end of after throwing advice....
java.lang.ArithmeticException: Not eligible for exam

Problem Statements 6: Write a program to demonstrate Spring AOP –pointcuts.

Solution:

Operation_pc.java

```
package bvimit.edu;
publicclass Operation_pc {

        publicvoid msg() {System.out.println("method 1");}
        publicint m() {System.out.println("method 2 with return");return 2;}
        publicint k() {System.out.println("method 3 with return");return 3;}
        }
```

Aopdata_pc.java

```
package bvimit.edu;
import  org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Pointcut;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
@Aspect
public class Aopdata_pc {

    @Pointcut("execution(int Operation.*(..))")
    public void p(){}

    @After("p()")
    public void myadvice(JoinPoint jp)
    {
        System.out.println("After advice");
    }
    @Pointcut("execution(* Operation.*(..))")
```

```
	public void i(){}

	@Before("i()")
	public void myadvice1(JoinPoint jp)
	{
		System.out.println("Before advice");
}
}
```

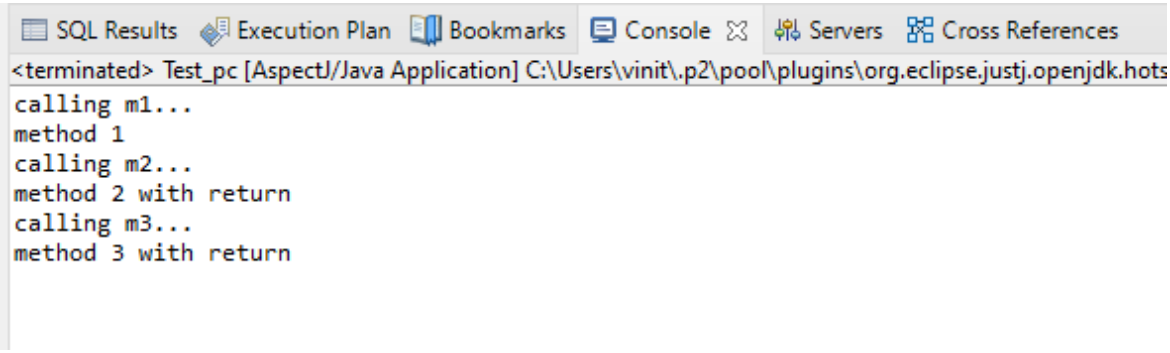
Test_pc.java

```
package bvimit.edu;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

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

ApplicationContext context = new ClassPathXmlApplicationContext("aopctx_pc.xml");
		Operation_pc e=(Operation_pc)context.getBean("opBean");
		System.out.println("calling m1...");
		e.msg();
		System.out.println("calling m2...");
		e.m();
		System.out.println("calling m3...");
		e.k();
		}
}
```

aopctx_pc.xml

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

<bean id="opBean" class="bvimit.edu.Operation_pc"></bean>

<bean id="trackMyBean" class="bvimit.edu.Aopdata_pc"></bean>

<bean class="org.springframework.aop.aspectj.annotation.AnnotationAwareAspectJAutoProxyCreator"></bean>

</beans>
```

Output:



Spring JDBC

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

Solution :

Movie1.java

```
package org.me;

publicclass Movie1 {

	int mid;
	String title;
	String actor;
	public Movie1(int mid, String title, String actor) {
		super();
		this.mid = mid;
		this.title = title;
		this.actor = actor;
	}
	public Movie1() {
		super();
		// TODO Auto-generated constructor stub
	}
	publicint getMid() {
		return mid;
	}
	publicvoid setMid(int mid) {
		this.mid = mid;
	}
	public String getTitle() {
		return title;
	}
	publicvoid setTitle(String title) {
		this.title = title;
	}
	public String getActor() {
		return actor;
	}
	publicvoid setActor(String actor) {
		this.actor = actor;
	}


}
```

MovieDAO.java

```
package org.me;

import org.springframework.jdbc.core.*;
publicclass MovieDAO {
JdbcTemplate jdbcTemplate;

publicvoid setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
}
publicint insMovie(Movie1 m1)
{
        String insSql="insert into mymovies1    values("+m1.getMid()+",'"+m1.getTitle()+"','"+m1.getActor()+"')";

return jdbcTemplate.update(insSql);
}

publicint updateMovie(Movie1 m1){
   String query="update mymovies1 set title='"+m1.getTitle()+"',actor='"+m1.getActor()+"' where mid='"+m1.getMid()+"'
";
return jdbcTemplate.update(query);
}

publicint deleteMovie(Movie1 m1){
  String query="delete from mymovies1 where mid='"+m1.getMid()+"' ";
return jdbcTemplate.update(query);
}
}
```

appctx.xml

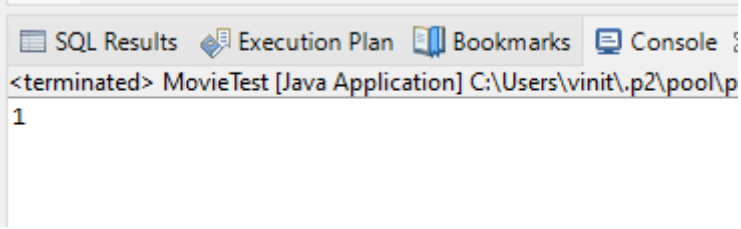
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName"value="org.postgresql.Driver" />
<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />
<property name="username" value="postgres" />
<property name="password" value="admin" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>

<bean id="mymovie" class="org.me.MovieDAO">
```

```
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean></beans>
```

MovieTest.java

```
package org.me;

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


public class MovieTest {
	private static ApplicationContext appCon;


	public static void main(String[] args) {
		// TODO Auto-generated method stub

		appCon = new ClassPathXmlApplicationContext("appctx.xml");
		MovieDAO m1=(MovieDAO)appCon.getBean("mymovie");

		//insert query
		Movie1 t1=new Movie1(10,"Mirzapur","P");
		System.out.println(m1.insMovie(t1));

		//update query

		//int status=m1.updateMovie(new Movie1(10,"war","hritik"));
	// System.out.println(status);


		//delete

		// Movie1 t2=new Movie1();
	  //t2.setMid(5);
	  //int status=m1.deleteMovie(t2);
	 // System.out.println(status);


	}

}
```

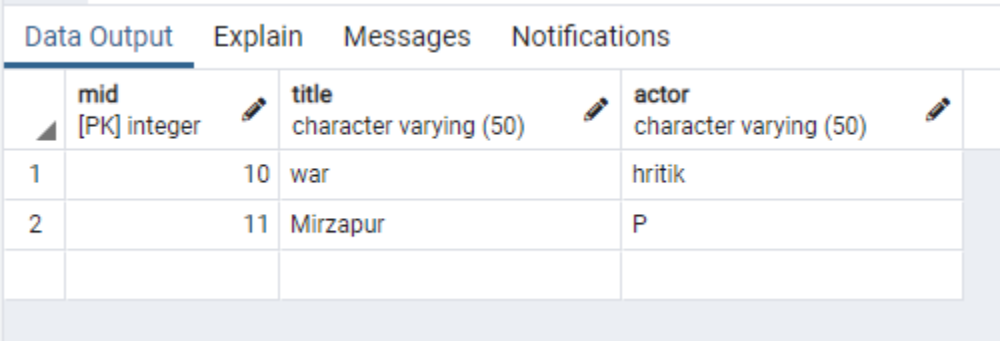
Output :



Database :

```
CREATE TABLE mymovies1
(
mid int,
title varchar(50),
actor varchar(50),
PRIMARY KEY (mid)
);
```

Final Table After Execution :



	mid [PK] integer	title character varying (50)	actor character varying (50)
1	10	war	hritik
2	11	Mirzapur	P

Problem Statement 2 : Write a program to demonstrate PreparedStatement in Spring JdbcTemplate.

Solution :

Movie1.java

```
package org.me;

public class Movie1 {

	int mid;
	String title;
	String actor;
	public Movie1(int mid, String title, String actor) {
		super();
		this.mid = mid;
		this.title = title;
		this.actor = actor;
	}
	public Movie1() {
		super();
	}
	public int getMid() {
		return mid;
	}
	public void setMid(int mid) {
		this.mid = mid;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public String getActor() {
		return actor;
	}
	public void setActor(String actor) {
		this.actor = actor;
	}

}
```

MovieDAO1.java

```
package org.me;

import java.sql.PreparedStatement;

import java.sql.SQLException;


import org.springframework.dao.DataAccessException;
```

```
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCallback;
public class MovieDAO1 {
	JdbcTemplate jdbcTemplate;

	public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
		 this.jdbcTemplate = jdbcTemplate;
	}
	public Boolean saveMovieByPreparedStatement(final Movie1 e){
	   String query="insert into movies values(?,?,?)";
	   return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){
	   @Override
	   public Boolean doInPreparedStatement(PreparedStatement ps)
		  throws SQLException, DataAccessException {
	      ps.setInt(1,e.getMid());
	      ps.setString(2,e.getTitle());
	      ps.setString(3,e.getActor());
	      return ps.execute();
	   }
	   });
	}
}
```

appctx1.java

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

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="org.postgresql.Driver" />
<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />
<property name="username" value="postgres" />
<property name="password" value="pass" />
</bean>
```

```
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>

<bean id="mymovie" class="org.me.MovieDAO1">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

MovieTest1.java

```
package org.me;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MovieTest1 {


	private static ApplicationContext appCon;

	public static void main(String[] args) {

		// TODO Auto-generated method stub

		appCon = new ClassPathXmlApplicationContext("appctx1.xml");

		MovieDAO1 m1=(MovieDAO1)appCon.getBean("mymovie");

	  m1.saveMovieByPreparedStatement(new Movie1(5,"Bhaijaan","Slemon"));

	}

}
```

Output :

Data Output | Explain | Messages | Notifications

	mid [PK] integer	title character varying (50)	actor character varying (50)
1	10	war	hritik
2	11	Mirzapur	P
3	4	Inception	Cobb
4	5	Bhaijaan	Slemon

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

Solution :

Movie2.java

```
package org.me;

public class Movie2 {

	int mid;
	String title;
	String actor;
	public int getMid() {
		return mid;
	}
	public void setMid(int mid) {
		this.mid = mid;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public String getActor() {
		return actor;
	}
	public void setActor(String actor) {
		this.actor = actor;
	}
	public String toString(){
	   return mid+""+title+""+actor;

}
}
```

MovieDAO2.java

```
package org.me;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import org.springframework.dao.DataAccessException;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.ResultSetExtractor;
```

```
public class MovieDAO2 {
	JdbcTemplate jdbcTemplate;

	public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
		this.jdbcTemplate = jdbcTemplate;
	}
	public List<Movie2> getAllMovie(){
		return jdbcTemplate.query("select * from mymovies1",new ResultSetExtractor<List<Movie2>>(){
		  @Override
		   public List<Movie2> extractData(ResultSet rs) throws SQLException,
		       DataAccessException {

		     List<Movie2> list=new ArrayList<Movie2>();
		     while(rs.next()){
		         Movie2 e=new Movie2();
		         e.setMid(rs.getInt(1));
		         e.setTitle(rs.getString(2));
		         e.setActor(rs.getString(3));
		         list.add(e);
		     }
		     return list;
		     }
		  });
		}
}
```

appctx2.java

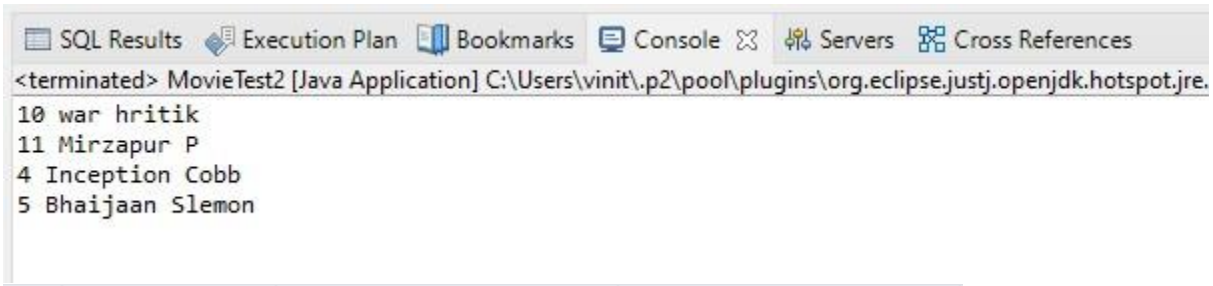
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
```

```
<property name="driverClassName" value="org.postgresql.Driver" />
<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />
<property name="username" value="postgres" />
<property name="password" value="password" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>

<bean id="mymovie" class="org.me.MovieDAO2">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

MovieTest2.java

```
package org.me;

import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MovieTest2 {

        private static ApplicationContext appCon;

        public static void main(String[] args) {

                appCon = new ClassPathXmlApplicationContext("appctx2.xml");

                MovieDAO2 m1=(MovieDAO2)appCon.getBean("mymovie");

                List<Movie2> list=m1.getAllMovie();


          for(Movie2 e:list)

              System.out.println(e);

         }

}
```

Output :



	mid [PK] integer	title character varying (50)	actor character varying (50)
1	10	war	hritik
2	11	Mirzapur	P
3	4	Inception	Cobb
4	5	Bhaijaan	Slemon

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

Solution :

Movie3.java

```
package org.me;

public class Movie3 {

	int mid;
	String title;
	String actor;
	public Movie3(int mid, String title, String actor) {
		super();
		this.mid = mid;
		this.title = title;
		this.actor = actor;
	}

	public Movie3() {
		super();
		// TODO Auto-generated constructor stub
	}
	public int getMid() {
		return mid;
	}
	public void setMid(int mid) {
		this.mid = mid;
	}
	public String getTitle() {
		return title;
	}
	public void setTitle(String title) {
		this.title = title;
	}
	public String getActor() {
		return actor;
	}
	public void setActor(String actor) {
		this.actor = actor;
	}
}
```

MovieDAO3.java

```
package org.me;


import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;


import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.RowMapper;

public class MovieDAO3 {

	JdbcTemplate jdbcTemplate;

	public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {

		this.jdbcTemplate = jdbcTemplate;

	}

	public List<Movie2> getAllEmployeesRowMapper(){

		 return jdbcTemplate.query("select * from mymovies1",new RowMapper<Movie2>(){

		   @Override

		   public Movie2 mapRow(ResultSet rs, int rownumber) throws SQLException {

			   Movie2 e=new Movie2();

		      e.setMid(rs.getInt(1));

		      e.setTitle(rs.getString(2));

		      e.setActor(rs.getString(3));

		      return e;

		   }

		   });

		}

}
```

appxtx3.xml

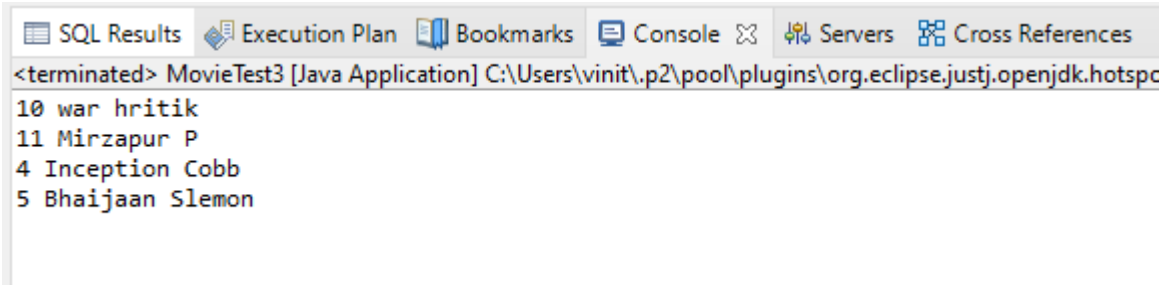
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="org.postgresql.Driver" />
<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />
<property name="username" value="postgres" />
<property name="password" value="password" />
</bean>

<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>

<bean id="mymovie" class="org.me.MovieDAO3">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```

MovieTest3.java

```
package org.me;


import java.util.List;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MovieTest3 {

	private static ApplicationContext appCon;

	public static void main(String[] args) {

		appCon = new ClassPathXmlApplicationContext("appctx3.xml");

		MovieDAO3 m1=(MovieDAO3)appCon.getBean("mymovie");

		List<Movie2> list=m1.getAllEmployeesRowMapper();


		   for(Movie2 e:list)

		      System.out.println(e);

	}
```

}

Output :



	mid [PK] integer	title character varying (50)	actor character varying (50)
1	10	war	hritik
2	11	Mirzapur	P
3	4	Inception	Cobb
4	5	Bhaijaan	Slemon

Spring Boot and RESTFUL Web Services

Problem Statement 1 : Write a program to create a simple Spring Boot application that prints a message.

Solution :

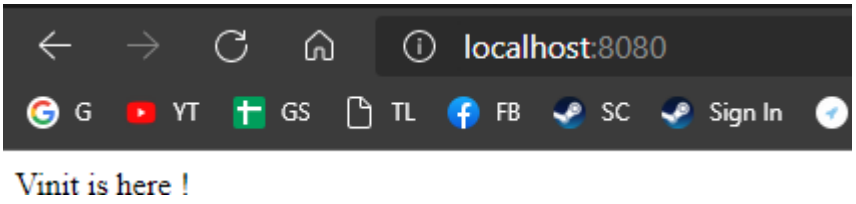
BoothelloApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class BoothelloApplication {

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

}
```

HelloWorldController.java

```
package com.example.demo;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class HelloWorldController {
	@RequestMapping("/")
	public String hello()
	{
		return "Vinit is here !";
	}
}
```

Output :



Problem Statement 2 : Write a program to demonstrate RESTful Web Services with spring boot

Solution :

HelloWorldBean.java

```
package com.example.demo;

publicclass HelloWorldBean {

	public String message;
	//constructor of HelloWorldBean
	public HelloWorldBean(String message)
	{
	this.message=message;
	}
	//generating getters and setters
	public String getMessage()
	{
	return message;
	}
	publicvoid setMessage(String message)
	{
	this.message = message;
	}
	@Override
	//generate toString
	public String toString()
	{
	return String.format ("HelloWorldBean [message=%s]", message);
	}
}
```

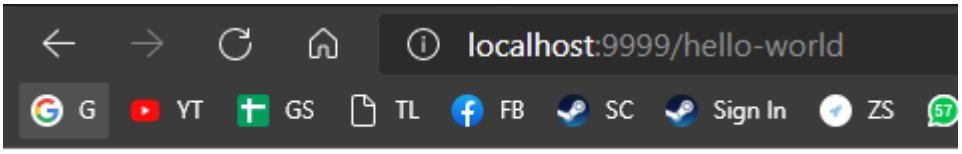
HelloWorldController.java

```
package com.example.demo;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
//Controller
@RestController
public class HelloWorldController
{
//using get method and hello-world as URI
	@GetMapping(path="/hello-world")
public String helloWorld()
{
return "Vinit is here!";
}
@GetMapping(path="/hello-world-bean")
public HelloWorldBean helloWorldBean()
{
return new HelloWorldBean("Kaise ho? xD"); //constructor of HelloWorldBean  }  }
```

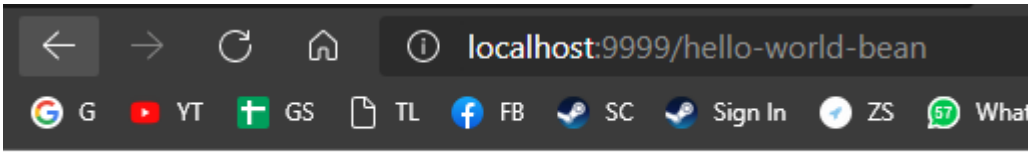
RestfulwebserviceApplication.java

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class RestfulwebserviceApplication {

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

}
```

Output :



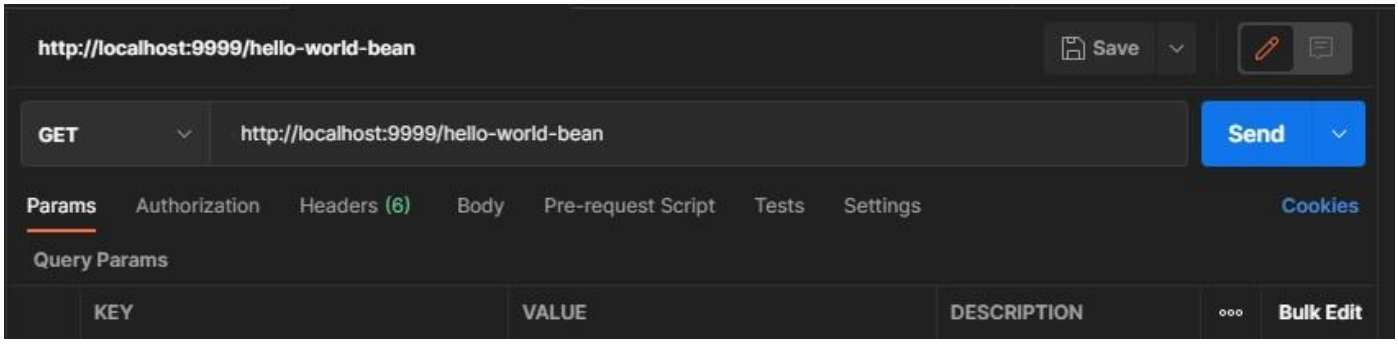
Vinit is here!

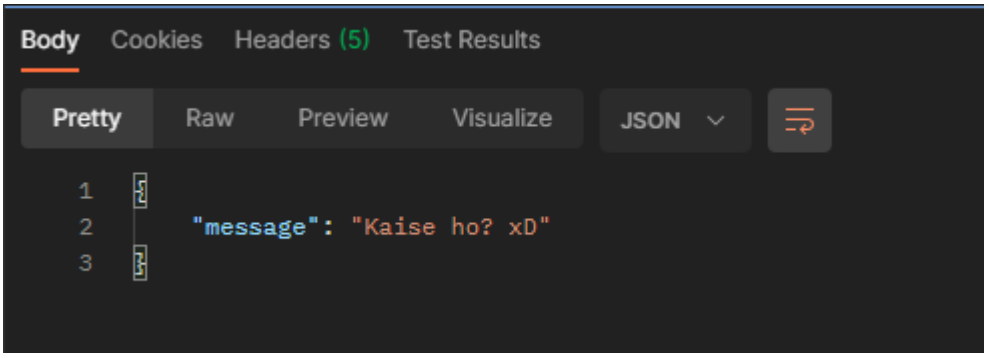


{"message":"Kaise ho? xD"}

Testing API with PostMan.

EndPoint : http://localhost:9999/hello-world-bean




Body
Cookies
Headers (5)
Test Results
Pretty
Raw
Preview
Visualize
JSON
1
2
3
"message": "Kaise ho? xD"