PROGRAMS BASED ON JAVA GENERICS

Practical No.:01

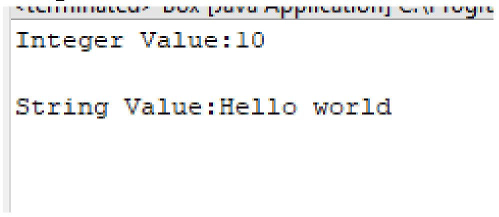
Program No.:01

Aim:-Write a Java Program to demonstrate generic class

Code:-

```
package gen_pro1;
import java.io.*;
import java.util.*;
public class Box<T>
 {
        private T t;
        public void add(T t)
        {
                this.t=t;
        }
        public T get ()
        {
                return t;
        }
         public static void main (String [] args)
        {
                 Box<Integer>integerBox=new Box<Integer>();
                 Box<String>stringBox=new Box<String>();
                 integerBox.add(new Integer (10));
                 stringBox.add(new String ("Hello world"));
                System.out.printf("Integer Value:%d\n\n",integerBox.get());
                System.out.printf("String Value:%s\n",stringBox.get());
        }
}
```

Output:



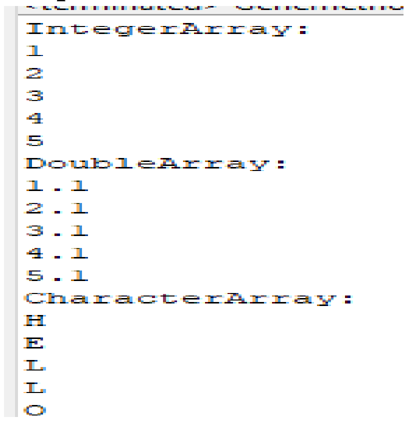
```
Integer Value:10

String Value:Hello world
```

PROGRAMS BASED ON JAVA GENERICS

Program No.:02
Aim:Write a Java program to Demonstrate Generic Method
Code:-

```
package gen_pro1;
import java.io.*;
import java.util.*;
public class Genemethod
{
        public static<E> void printArray(E[]arr)
        {
                for(E element :arr)
                {
                        System.out.printf("%S",element);
                        System.out.println();
                }
        }
        public static void main(String[]args)
        {
                Integer[] intarr= {1,2,3,4,5};
                Double[] darr= {1.1,2.1,3.1,4.1,5.1};
                Character[]carr= {'H','e','l','l','o'};
                System.out.println("IntegerArray: ");
                printArray(intarr);
                System.out.println("DoubleArray: ");
                printArray(darr);
                System.out.println("CharacterArray: ");
                printArray(carr);
        }
}
```

Output:



```
IntegerArray:
1
2
3
4
5
DoubleArray:
1.1
2.1
3.1
4.1
5.1
CharacterArray:
H
E
L
L
O
```

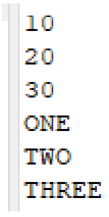
Roll no.: 13 **Jyoti Gupta**

PROGRAMS BASED ON JAVA GENERICS

Program No.:03
Aim: Write a java program to demonstrate wildcard of unknown type
Code:

```
package gen_pro1;
import java.util.*;
public class Unknown
{
        static void processElements(ArrayList<?>a)
        {
                for(Object element:a)
                {
                        System.out.println(element);
                }
        }
        public static void main(String[] args)
        {
                ArrayList<Integer>a1=new ArrayList<>();
                a1.add(10);
                a1.add(20);
                a1.add(30);
                processElements(a1);
                ArrayList<String>a2=new ArrayList<>();
                a2.add("ONE");
                a2.add("TWO");
                a2.add("THREE");
                processElements(a2);
        }

}
```

Output:



```
10
20
30
ONE
TWO
THREE
```

PROGRAMS BASED ON JAVA GENERICS

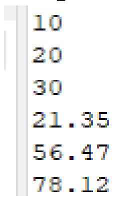
Program No.:04

Aim:Write a Java Program to demonstrate Wildcard Arguments with an UpperBound.

Code:

```
package gen_pro1;
import  java.util.*;
public class Upperbound
{
	static void processElements(ArrayList<? extends Number> a)
	{
		for(Object element:a)
		{
			System.out.println(element);
		}
	}
	public static void main(String[] args)
	{
		ArrayList<Integer> a1=new ArrayList<>();
		a1.add(10);
		a1.add(20);
		a1.add(30);
		processElements(a1);
		ArrayList<Double> a2=new ArrayList<>();
		a2.add(21.35);
		a2.add(56.47);
		a2.add(78.12);
		processElements(a2);
		ArrayList<String>a3=new ArrayList<>();
		a3.add("One");
		a3.add("Two");
		a3.add("Three");
	}
}
```

Output:



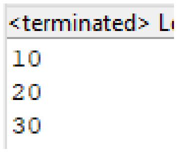
```
10
20
30
21.35
56.47
78.12
```

PROGRAMS BASED ON JAVA GENERICS

Program 5:
Aim:Write a Java Program to demonstrate Wildcard Arguments with an LowerBound.
Code:

```
package gen_pro1;
import java.util.*;
public class Lowerbound
{
	static void processElements(ArrayList<? super Integer>a)
	{
		for(Object element:a)
		{
			System.out.println(element);
		}
	}
	public static void main(String args[])
	{
		ArrayList<Integer> a1=new ArrayList<>();
		a1.add(10);
		a1.add(20);
		a1.add(30);
		processElements(a1);
		ArrayList<Double>a2 = new ArrayList<>();
		a2.add(21.35);
		a2.add(56.47);
		a2.add(78.12);

	}
}
```

Output:



```
<terminated> L
10
20
30
```

PROGRAMS BASED ON JAVA GENERICS

Practical No:02

Program No.:01

Aim:- Write a java program to create list and demonstrate all operations of List.

a) **Add element**
b) **Appending list elements**
c) **Clear/empty list**
d) **Size of list**
e) **Updating elements in list using set**
f) **Extracting a portion of list**
g) **Removing elements from list**
h) **Searching for an element in a list**
i) **Sorting a list**
j) **Copying elements from one list into another**
k) **Shuffling elements in a list**
l) **Reversing elements in a list**

Code:

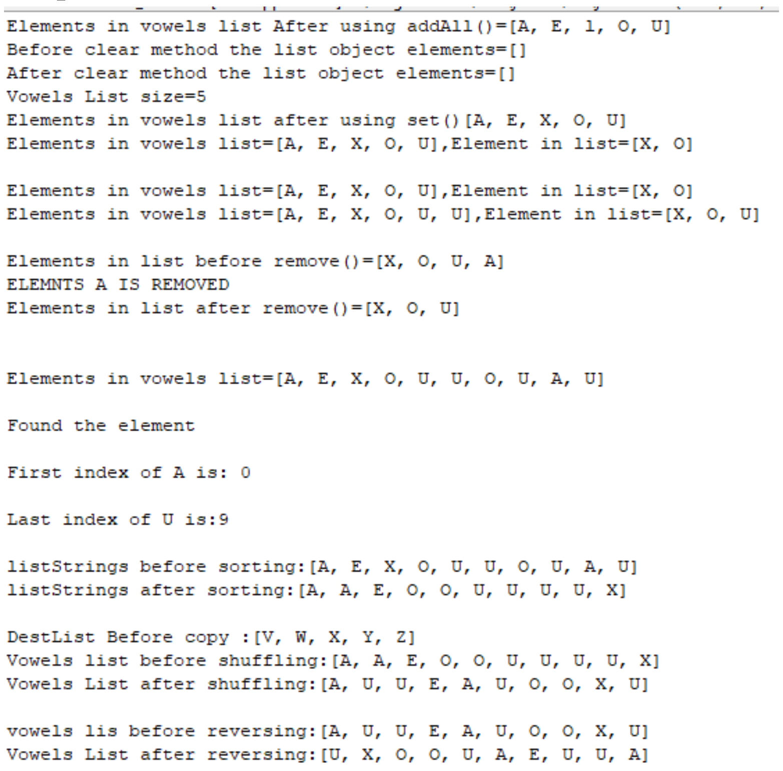
```
import java.util.*;
public class List_interface {
        public static void main(String args[])
        {
                List<String> vowels=new ArrayList<String>(25);
                                //add example

                vowels.add("A");
                vowels.add("l");
                //let's insert E between A and l
                vowels.add(1, "E");
                List<String>list=new ArrayList<String>();
                vowels.add("O");
                vowels.add("U");
                //appending list elements to letters
                vowels.addAll(list);
          System.out.println("Elements in vowels list After using addAll()="+vowels);
          //clear example to empty list
          System.out.println("Before clear method the list object elements="+list);
          list.clear();
          System.out.println("After clear method the list object elements="+list);
          //size example
          System.out.println("Vowels List size="+vowels.size());
```

```
 //update elements in a list using set
 vowels.set(2, "X");
 System.out.println("Elements in vowels list after using set()"+vowels);
 //Extracting a portion of a list
 /* The sub list(fromindex,toindex)allows us to get portion of the list between the
 specified fromindex(inclusive) and toindex(exclusive).*
 list=vowels.subList(2, 4);
 System.out.println("Elements in vowels list="+vowels+",Element in list="+list);
 System.out.println();
 vowels.set(0, "A");
 System.out.println("Elements in vowels list="+vowels+",Element in list="+list);
 list.add("U");
 System.out.println("Elements in vowels list="+vowels+",Element in list="+list);
 System.out.println();
 list.add("A");
 //removing elements from list
 System.out.println("Elements in list before remove()="+list);
 if(list.remove("A")) {
        System.out.println("ELEMNTS A IS REMOVED");
 }else {
        System.out.println("There is no such element");
 }
 System.out.println("Elements in list after remove()="+list);
 System.out.println();
 vowels.add("O");
 vowels.add("U");
 vowels.add("A");
 vowels.add("U");
 System.out.println();
 System.out.println("Elements in vowels list="+vowels);
 System.out.println();
 //searching for an element in a list
if(vowels.contains("U"))
 {
          System.out.println("Found the element");
 }else {
          System.out.println("There is no such element");
 }
System.out.println();
int firstIndex=vowels.indexOf("A");
```

```
  System.out.println("First index of A is: "+firstIndex);
    System.out.println();
    int lastindex=vowels.lastIndexOf("U");
    System.out.println("Last index of U is:"+lastindex);
    //sorting
    System.out.println();
    System.out.println("listStrings before sorting:"+vowels);
    Collections.sort(vowels);
    System.out.println("listStrings after sorting:"+vowels);
    System.out.println();
//copying elements from one list into another
    List<String> sourceList=new ArrayList<String>();
    sourceList.add("A");
    sourceList.add("B");
    sourceList.add("C");
    sourceList.add("D");
    List<String> destList=new ArrayList<String>();
    destList.add("V");
    destList.add("W");
    destList.add("X");
    destList.add("Y");
    destList.add("Z");
    System.out.println("DestList Before copy :"+destList);
    Collections.copy(destList, sourceList);
    //shuffling elements in list
     System.out.println("Vowels list before shuffling:"+vowels);
     Collections.shuffle(vowels);
     System.out.println("Vowels List after shuffling:"+vowels);
     System.out.println();
     //reversing an array
     System.out.println("vowels lis before reversing:"+vowels);
     Collections.reverse(vowels);
     System.out.println("Vowels List after reversing:"+vowels);
     System.out.println();
     }
  }
```

PROGRAMS BASED ON JAVA GENERICS

Output:-



```
Elements in vowels list After using addAll()=[A, E, l, O, U]
Before clear method the list object elements=[]
After clear method the list object elements=[]
Vowels List size=5
Elements in vowels list after using set()[A, E, X, O, U]
Elements in vowels list=[A, E, X, O, U],Element in list=[X, O]

Elements in vowels list=[A, E, X, O, U],Element in list=[X, O]
Elements in vowels list=[A, E, X, O, U, U],Element in list=[X, O, U]

Elements in list before remove()=[X, O, U, A]
ELEMNTS A IS REMOVED
Elements in list after remove()=[X, O, U]


Elements in vowels list=[A, E, X, O, U, U, O, U, A, U]

Found the element

First index of A is: 0

Last index of U is:9

listStrings before sorting:[A, E, X, O, U, U, O, U, A, U]
listStrings after sorting:[A, A, E, O, O, U, U, U, U, X]

DestList Before copy :[V, W, X, Y, Z]
Vowels list before shuffling:[A, A, E, O, O, U, U, U, U, X]
Vowels List after shuffling:[A, U, U, E, A, U, O, O, X, U]

vowels lis before reversing:[A, U, U, E, A, U, O, O, X, U]
Vowels List after reversing:[U, X, O, O, U, A, E, U, U, A]
```

PROGRAMS BASED ON JAVA GENERICS

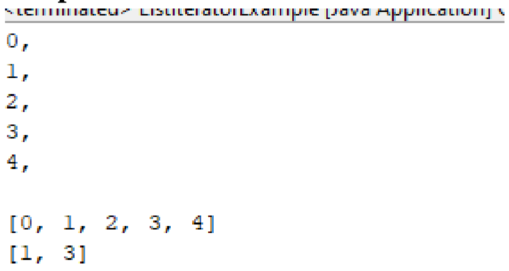
Program No:- 02

Aim: write a java program to create List containing list of items using ListIterator

Code:

```
import java.util.*;
public class ListIteratorExample
{
        public static void main(String args[])
        {
                List<Integer>list=new ArrayList<>();
                for(int i=0;i<5;i++)
                        list.add(i);
                Iterator<Integer>iterator=list.iterator();
                //simple iteration
                while(iterator.hasNext())
                {
                        int i= (int) iterator.next();
                        System.out.println(i+",");
                }
                System.out.println("\n"+list);
                //modifications of list using iterator
                iterator=list.iterator();
                while (iterator.hasNext())
                {
                int x=(int)iterator.next();
                if(x%2==0)
                iterator.remove();
                }
                System.out.println(list)
        }}
```

Output:



```
<terminated> ListIteratorExample [Java Application]
0,
1,
2,
3,
4,

[0, 1, 2, 3, 4]
[1, 3]
```

Programs Based On Set Interface

Practical No:03

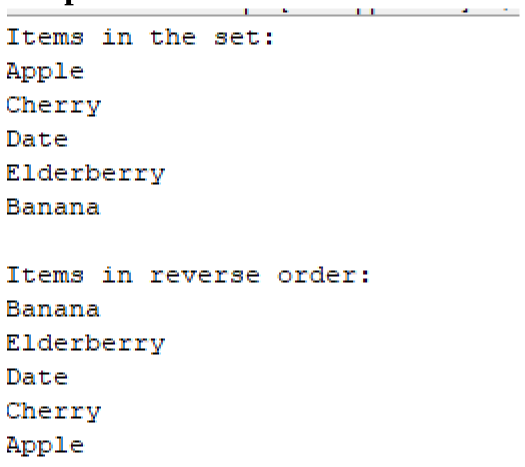
Program No:- 01

Aim: Write a JAVA program to create a set containing list of items as type string and print the items in the list using iterator interface. Also print the list in reverse/ backward direction.

Code:

```
import java.util.*;
public class setExample {
        public static void main(String[] args) {
                Set<String> itemSet=new HashSet<>();
                itemSet.add("Apple");
                itemSet.add("Banana");
                itemSet.add("Cherry");
                itemSet.add("Date");
                itemSet.add("Elderberry");
                System.out.println("Items in the set:");
                Iterator<String>iterator=itemSet.iterator();
                while (iterator.hasNext())
                {
                        System.out.println(iterator.next());
                }
                List<String>itemList=new ArrayList<>(itemSet);
                System.out.println("\nItems in reverse order:");
                for(int i= itemList.size() -1;i>=0;i--) {
                        System.out.println(itemList.get(i));
                }}
}
```

Output:



```
Items in the set:
Apple
Cherry
Date
Elderberry
Banana

Items in reverse order:
Banana
Elderberry
Date
Cherry
Apple
```

Programs Based On Set Interface

Program No:02

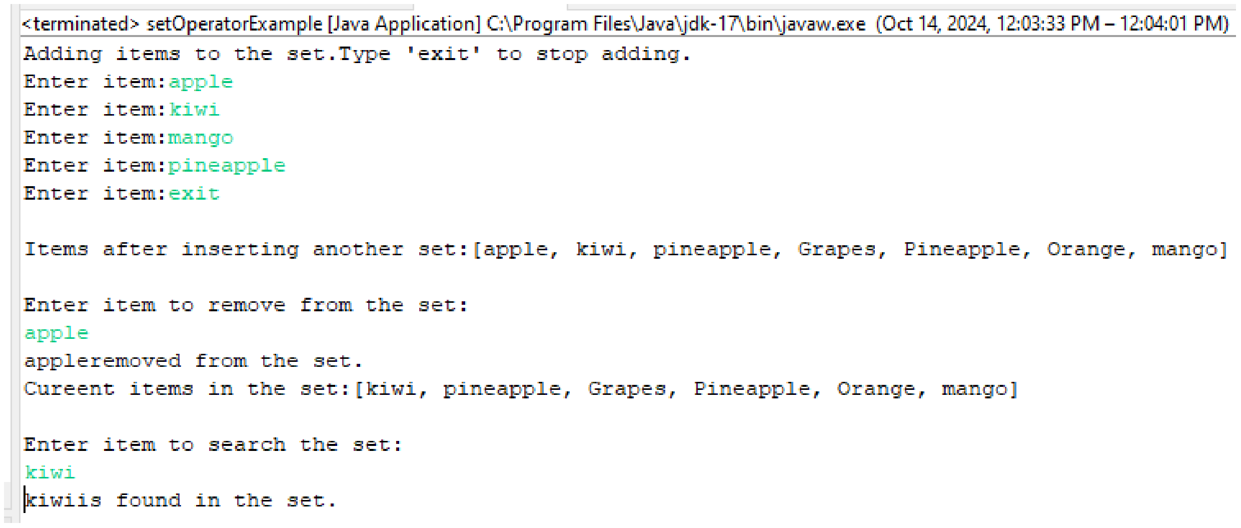
Aim: Write a JAVA program using set Interface containing List of items and perform the following Operations.

- **a) Add items in the set**
- **b) Insert items of one set into other set**
- **c) Remove items from the set**
- **d) Search the specified item in the set**

Code:

```
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;
public class setOperatorExample {
        public static void main(String[] args) {
                Scanner scanner=new Scanner(System.in);

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

                System.out.println("Adding items to the set.Type 'exit' to stop adding.");
                while(true) {
                        System.out.print("Enter item:");
                        String item=scanner.nextLine();
                        if(item.equalsIgnoreCase("exit")) {
                                break;
                        }
                   itemset.add(item);
                }
                Set<String> anotherSet=new HashSet<>();
                anotherSet.add("Grapes");
                anotherSet.add("Orange");
                anotherSet.add("Pineapple");

                itemset.addAll(anotherSet);
                System.out.println("\nItems after inserting another set:"+ itemset);

                System.out.println("\nEnter item to remove from the set:");
                String itemToRemove= scanner.nextLine();
                if(itemset.remove(itemToRemove)) {
                        System.out.println(itemToRemove +"removed from the set.");
                }else {
                        System.out.println(itemToRemove +"not found in the set.");
                }
                System.out.println("Cureent items in the set:" +itemset);

                System.out.println("\nEnter item to search the set:");
                String itemToSearch=scanner.nextLine();
                if(itemset.contains(itemToSearch)) {
                        System.out.println(itemToSearch + "is found in the set.");
                }else {
                        System.out.println(itemToSearch+"is not found in the set.");
                }
                scanner.close();

        }

}
```

Output:



```
<terminated> setOperatorExample [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (Oct 14, 2024, 12:03:33 PM – 12:04:01 PM)
Adding items to the set.Type 'exit' to stop adding.
Enter item:apple
Enter item:kiwi
Enter item:mango
Enter item:pineapple
Enter item:exit

Items after inserting another set:[apple, kiwi, pineapple, Grapes, Pineapple, Orange, mango]

Enter item to remove from the set:
apple
appleremoved from the set.
Cureent items in the set:[kiwi, pineapple, Grapes, Pineapple, Orange, mango]

Enter item to search the set:
kiwi
kiwiis found in the set.
```

PROGRAMS BASED ON MAP INTERFACE

Practical No.:04

Program No.:01

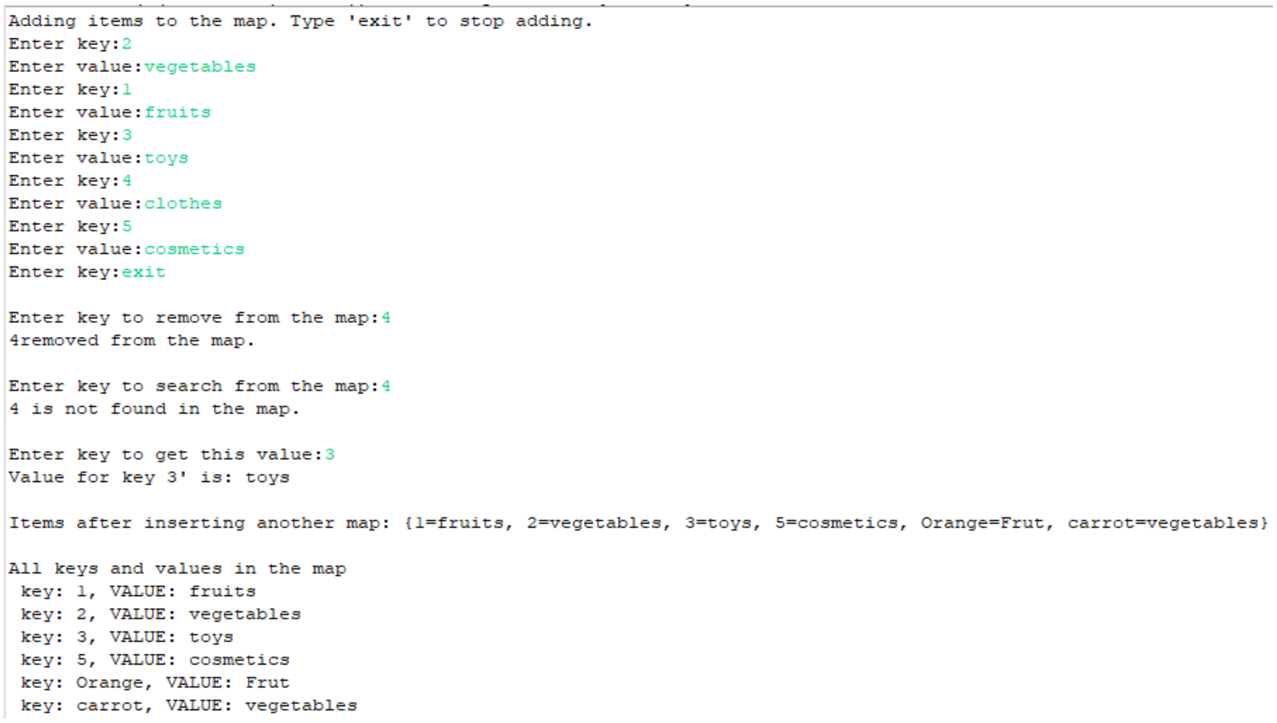
Aim:Write down a Java Program using Map Interface containing list of items having keys and associated values and perform the following operations:

a. **Add items in the map**
b. **Remove items from the map**
c. **Search specific key from the key**
d. **Insert map elements of one map into another map.**
e. **Insert map elements of one map into another map.**
f. **Print all keys and values of the map**

Code:

```
import java.util.HashMap;
import  java.util.Map;
import java.util.Map.Entry;
import java.util.Scanner;
public class MapOpeartionsExample {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
                //Create a Map to store items
                Map<String, String> itemMap= new HashMap<>();
                //a.Add items to the map
                System.out.println("Adding items to the map. Type 'exit' to stop adding.");
                while (true) {
                        System.out.print("Enter   key:");
                        String  key=  scanner.nextLine();
                        if(key.equalsIgnoreCase("exit"))
                        {break;}
                        System.out.print("Enter value:");
                        String value= scanner.nextLine();
                        itemMap.put(key, value);}
                //b.remove items form the map
                System.out.print("\nEnter key to remove from the map:");
                String keyToRemove= scanner.nextLine();
                if(itemMap.remove(keyToRemove)!=null) {
                        System.out.println(keyToRemove+"removed from the map.");
                        }else {
                                System.out.println(keyToRemove+" not found in the map.");}
                //d.Get value of the sepecific key in map
                System.out.print("\nEnter key to search from the map:");
                String keyToSearch= scanner.nextLine();
if(itemMap.containsKey(keyToSearch)) {
        System.out.println(keyToSearch+"removed from the map.");
        }else {
                System.out.println(keyToSearch+" is not found in the map.");}
System.out.print("\nEnter key to get this value:");
String keyToGetValue=scanner.nextLine();
String value=itemMap.get(keyToGetValue);
if(value !=null) {
        System.out.println("Value for key "+keyToGetValue+"' is: "+value);}else
        {System.out.println("key '"+keyToGetValue+"' not found.");}
//e.Insert elements of one map into another map
Map<String, String> anotherMap = new HashMap<>();
anotherMap.put("Orange","Frut");
anotherMap.put("carrot", "vegetables");
itemMap.putAll(anotherMap);
System.out.println("\nItems after inserting another map: "+itemMap);

//f.print all keys and values of the map
System.out.println("\nAll keys and values in the map");
for(Entry<String, String> entry:itemMap.entrySet())
{System.out.println(" key: "+entry.getKey()+", VALUE: "+entry.getValue());}
scanner.close();}}
```

Output:



```
Adding items to the map. Type 'exit' to stop adding.
Enter key:2
Enter value:vegetables
Enter key:1
Enter value:fruits
Enter key:3
Enter value:toys
Enter key:4
Enter value:clothes
Enter key:5
Enter value:cosmetics
Enter key:exit

Enter key to remove from the map:4
4removed from the map.

Enter key to search from the map:4
4 is not found in the map.

Enter key to get this value:3
Value for key 3' is: toys

Items after inserting another map: {1=fruits, 2=vegetables, 3=toys, 5=cosmetics, Orange=Frut, carrot=vegetables}

All keys and values in the map
 key: 1, VALUE: fruits
 key: 2, VALUE: vegetables
 key: 3, VALUE: toys
 key: 5, VALUE: cosmetics
 key: Orange, VALUE: Frut
 key: carrot, VALUE: vegetables
```

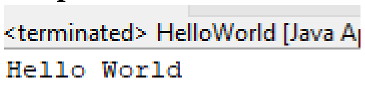
PROGRAMS BASED ON LAMBDA EXPRESSION

Practical No.:05

Program No.:01
Aim:-Write a Java Program using Lambda Expression to print “Hello World”.
Code:

```
interface Hello
{
	void sayHello();

}
public class HelloWorld {
	public static void main(String[] args)
	{
		Hello hello=()->System.out.println("Hello World");
		hello.sayHello();
	}

}
```

Output:-



```
<terminated> HelloWorld [Java A
Hello World
```

Roll no.:13 **Name: Jyoti Gupta**

PROGRAMS BASED ON LAMBDA EXPRESSION

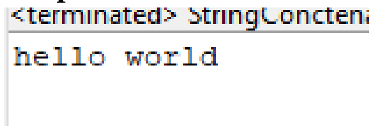
Program No.:02

Aim:-Write a Java Program using Lambda Expression to concatenate two strings.

Code:

```
interface Concatenate
{
	String join(String s1,String s2);
}

public class StringConctenate
{
	public static void main(String[] args)
	{
		Concatenate concat=(s1,s2)->s1+s2;
		System.out.println(concat.join("hello"," world"));
	}
}
```

Output:-



```
<terminated> StringConctena
hello world
```

Roll no.:13 **Name: Jyoti Gupta**

PROGRAMS BASED ON LAMBDA EXPRESSION

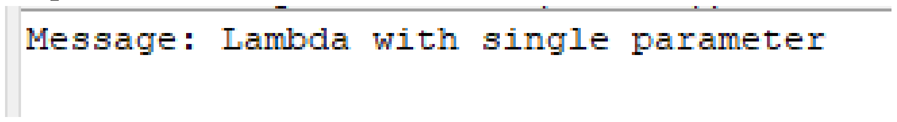
Program No.:03

Aim:-Write a Java Program using Lambda Expression with Single Parameter.

Code:

```
interface SingleParameter
{
	void display(String message);
}
public class SingleParameterExample
{
	public static void main (String[] args)
	{
		SingleParameter displayMessage=message->System.out.println("Message: 
"+message);
		displayMessage.display("Lambda with single parameter");
	}
}
```

Output:-



```
Message: Lambda with single parameter
```

PROGRAMS BASED ON LAMBDA EXPRESSION

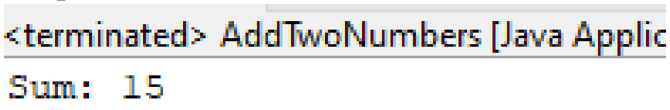
Program No.:04

Aim:-Write a Java Program using Lambda Expression with Multiple Parameter to add two numbers.

Code:

```
interface Add
{
	int sum(int a, int b);
}
public class AddTwoNumbers {
	public static void main(String[] args)
	{
		Add addition=(a,b)->a+b;
		System.out.println("Sum: "+addition.sum(5, 10));
	}

}
```

Output:



```
<terminated> AddTwoNumbers [Java Applic
Sum: 15
```

PROGRAMS BASED ON LAMBDA EXPRESSION

Program No.:05

Aim:-Write a Java Program using Lambda Expression to calculate following:-

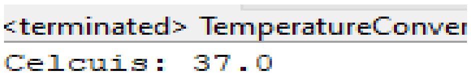
- **a. Convert Fahrenheit to Celcius.**
- **b. Convert Kilometers to Miles.**

a. Convert Fahrenheit to Celcius:

Code:

```
interface FahrenheitToCelcuis
{
	double convert(double fahrenheit);
}
public class TemperatureConversion
{
	public static void main (String[] args)
	{
		FahrenheitToCelcuis convert=f->(5.0/9)*(f-32);
		System.out.println("Celcuis: "+convert.convert(98.6));
	}
}
```

Output:-

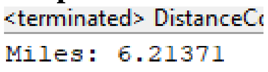


```
<terminated> TemperatureConver
Celcuis: 37.0
```

b. Convert Kilometers to Miles:Code:

```
interface KilometersToMiles{
	double convert(double km);
}
public class DistanceConversion
{
	public static void main(String[] args)
	{
		KilometersToMiles convert=km -> km*0.621371;
		System.out.println("Miles: "+convert.convert(10));
	}
}
```

Output:



```
<terminated> DistanceC
Miles: 6.21371
```

Roll no.:13 **Jyoti Gupta**

PROGRAMS BASED ON LAMBDA EXPRESSION

Program No.:06

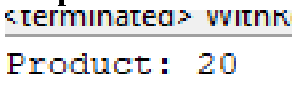
Aim:-Write a Java Program using Lambda Expression with and without return keyword.

6.1:-With return keyword

Code:

```
interface Multiply{
        int product(int a, int b);
}
public class WithReturn {
        public static void main(String[] args)
        {
                Multiply multiply=(a,b)->
                {
                        return a*b;
                };
                System.out.println("Product: "+multiply.product(5, 4));
        }
}
```

Output:



```
<terminated> WithR
Product: 20
```

Roll no.:13 **Jyoti Gupta**

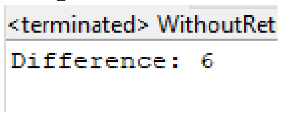
PROGRAMS BASED ON LAMBDA EXPRESSION

Program No.:06

6.1:-Without return keyword
Code:

```
interface Subtract
{
	int difference(int a,int b);
}
public class WithoutRetrun
{
	public static void main(String[] args)
	{
		Subtract subtract=(a,b)->a-b;
		System.out.println("Difference: "+subtract.difference(10, 4));
	}
}
```

Output:



```
<terminated> WithoutRet
Difference: 6
```

Programs based on web application development using JSP

Practical no. 6

Program no. 1:- jspDemo1

AIM : Write a jsp program to demonstrate directives declaration on jsp page.

Index.jsp

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-Tyoe" content="text/html; charset=UTF-8">
<title>Page Attribute </title>
</head>
<body>
<form action="directive.jsp">
<h1> Enter the value of n1 and n2: </h1>
Number1: <input type="number" name="n1"/><br/>
Number2: <input type="number" name="n2"/><br/>
<input type="submit"/>
</form>
</body>
</html>
```

error.jsp

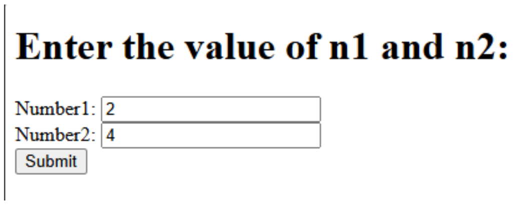
```
<%@ page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page isErrorPage="true" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-Type" content="text/html; charset=UTF-8">
<title>Page Attributes</title>
</head>
<body>
<h2> Value of n2 variable of zero(n/0 is infinity)</h2>
<h3> Sorry an exception occurred!</h3> <br/>
<h3> The Exception is: <%= exception  %></h3>
</body>
</html>
```

directive.jsp

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

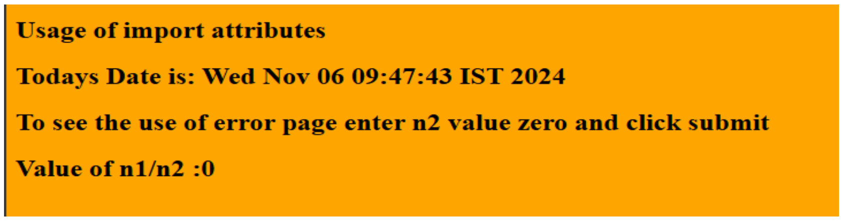
Programs based on web application development using JSP

```
<%@ page import="java.util.*" %>
<%@ page info="composed by DSATM" %>
<%@ page language="java" %>
<%@ page buffer="16kb" %>
<%@ page autoFlush="true" %>
<%@ page isThreadSafe="true" %>
<%@ page errorPage="error.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-Type" content="text/html; charset=UTF-8">
<title>Page Attributes</title>
</head>
<body bgcolor="orange">
<h2> Usage of import attributes</h2>
<h2> Todays Date is: <%=new Date() %></h2>
<h2> To see the use of error page enter n2 value zero and click submit</h2>
<%
int n1=Integer.parseInt(request.getParameter("n1")); int
n2=Integer.parseInt(request.getParameter("n2"));
%>
<h2> Value of n1/n2 :<%=n1/n2 %></h2>
</body>
</html>
```



Enter the value of n1 and n2:

Number1: 2
Number2: 4

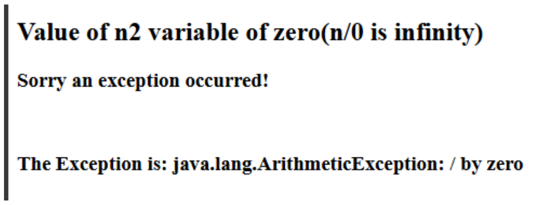


Usage of import attributes

Todays Date is: Wed Nov 06 09:47:43 IST 2024

To see the use of error page enter n2 value zero and click submit

Value of n1/n2 :0



Value of n2 variable of zero(n/0 is infinity)

Sorry an exception occurred!

The Exception is: java.lang.ArithmeticException: / by zero

Programs based on web application development using JSP

Program no. 2:- jspDemo2

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

Sonia.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"
%>
<%@ include file="header.html" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="ISO-8859-1">
    <title>JSP Demo Page</title>
</head>
<body>
    <h1>Enter the Message</h1>
    <form action="Response.jsp" method="post">
        Enter Message: <input type="text" value="" name="text1">
        <input type="submit" value="Submit" />
    </form>
</body>
</html>
<%@ include file="footer.html" %>
```

Response.jsp

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

1"%> <%@ include file="header.html" %>

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<% String s1=request.getParameter("text1"); %>
 <%=s1%>
</body>
```

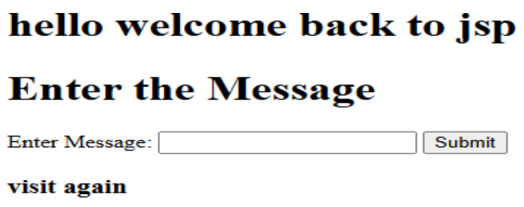
```
</html>
<%@include file="footer.html" %>
```

header.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        <h1> hello welcome back to jsp</h1>
</body>
</html>
```

footer.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
        <h3>visit again</h3>
</body>
</html>
```

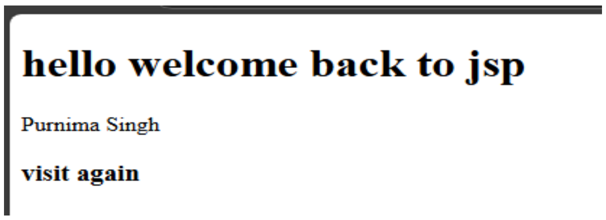


hello welcome back to jsp

Enter the Message

Enter Message: Submit

visit again



hello welcome back to jsp

Purnima Singh

visit again

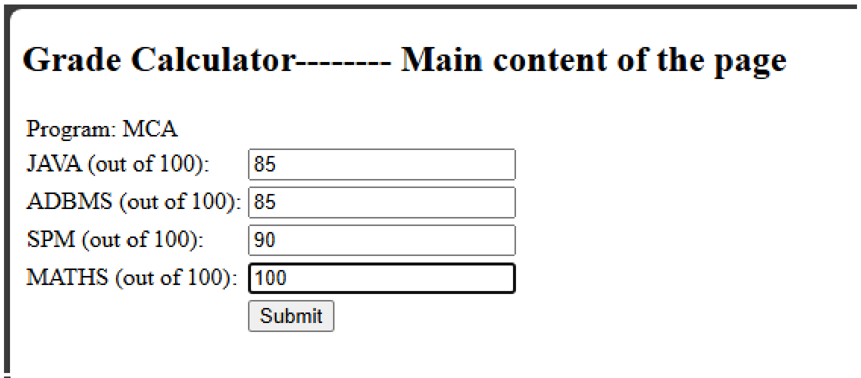
Programs based on web application development using JSP

Program no. 3 calculator

AIM: - Design Grade calculator using JSP which accepts Marks of subjects. Calculate average of marks and Display the grade of student based on average marks**.**

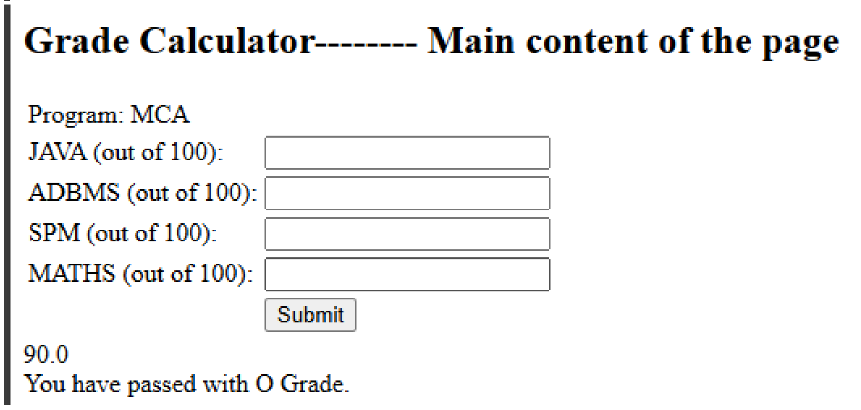
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-
1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>Grade Calculator-------- Main content of the page</h2>
<form action="" method="post">
<table>
  <tr><td>Program: MCA</td>
  <td></td></tr>
  <tr><td>JAVA (out of 100):</td>
  <td><input type="text" value="" name="text1"></td></tr>
  <tr><td>ADBMS (out of 100):</td>
  <td><input type="text" value="" name="text2"></td></tr>
  <tr><td>SPM (out of 100):</td>
  <td><input type="text" value="" name="text3"></td></tr>
  <tr><td>MATHS (out of 100):</td>
  <td><input type="text" value="" name="text4"></td></tr>
  <tr><td></td>
  <td><input type="submit" value="Submit" name="Submit" /></td></tr>
  <tr>
</table>
</form>
<%   try {
                        String res="";
int t1=Integer.parseInt(request.getParameter("text1")); int
t2=Integer.parseInt(request.getParameter("text2")); int
t3=Integer.parseInt(request.getParameter("text3")); int
t4=Integer.parseInt(request.getParameter("text4"));
     float r, av;       r=t1+t2+t3+t4;        av=r/4;
if(av>75 && av<=100){        res="You have
passed with O Grade."; }
     else if(av>60 && av<=75){        res="You
have passed with A Grade."; }
```

```
        else if(av>50 && av<=60){         res="You
have passed with B Grade."; }
        else if(av>40 && av<=50){         res="You
have passed with C Grade."; }
        else{
        res="You have Failed."; }
 %>
 <%= av%>
     <br>
       <%= res %>
     <% } catch(Exception e) { } %>
</body>
</html>
```



Grade Calculator-------- Main content of the page

Program: MCA	
JAVA (out of 100):	85
ADBMS (out of 100):	85
SPM (out of 100):	90
MATHS (out of 100):	100



Grade Calculator-------- Main content of the page

Program: MCA	
JAVA (out of 100):	
ADBMS (out of 100):	
SPM (out of 100):	
MATHS (out of 100):	

```
90.0
You have passed with O Grade.
```

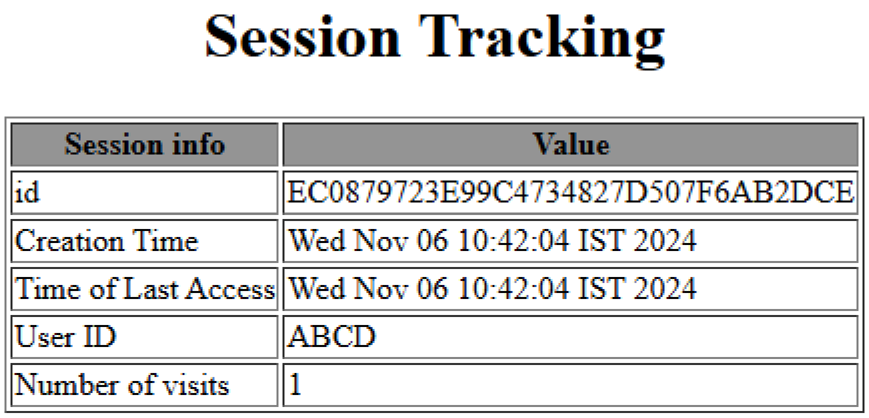
Programs based on web application development using JSP

Program no. 4 Session

AIM: Write a program in JSP to demonstrate session tracking technique.

Session.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>
<%@ page import = "java.io.*,java.util.*" %>
<%   // Get session creation time.
Date createTime = new Date(session.getCreationTime());    //
Get last access time of this Webpage.
Date lastAccessTime = new Date(session.getLastAccessedTime());
   String title = "Welcome Back to my website";
   Integer visitCount = new Integer(0);
   String visitCountKey = new String("visitCount");
   String userIDKey = new String("userID");
   String userID = new String("ABCD");
 // Check if this is new comer on your Webpage
  if (session.isNew() ){           title = "Welcome to my
website";     session.setAttribute(userIDKey,     userID);
session.setAttribute(visitCountKey,  visitCount);
}
visitCount = (Integer)session.getAttribute(visitCountKey);
visitCount = visitCount + 1;
userID = (String)session.getAttribute(userIDKey);
session.setAttribute(visitCountKey,  visitCount);
%>
<body>
<center>
    <h1>Session Tracking</h1>
</center>
<table border = "1" align = "center">
  <tr bgcolor = "#949494">
    <th>Session info</th>
    <th>Value</th>
  </tr>
  <tr>
  <td>id</td>
  <td><% out.print( session.getId()); %></td>
  </tr>
 <tr>
 <td>Creation Time</td>
 <td><% out.print(createTime); %></td>
 </tr>
  <tr>
  <td>Time of Last  Access</td>
  <td><% out.print(lastAccessTime); %></td>
  </tr>
  <tr>
  <td>User ID</td>
  <td><% out.print(userID); %></td>
   </tr>
   <tr>
   <td>Number of visits</td>
   <td><% out.print(visitCount); %></td>
   </tr>
    </table>
</body>
</html>
```



Session Tracking

Session info	Value
id	EC0879723E99C4734827D507F6AB2DCE
Creation Time	Wed Nov 06 10:42:04 IST 2024
Time of Last Access	Wed Nov 06 10:42:04 IST 2024
User ID	ABCD
Number of visits	1

Programs based on web application development using JSP

Program no. 5 Student Record

AIM : WAP to Insert records in Student Master.

PgADMIN

```
CREATE TABLE student(
    sname varchar(10),
    srollno varchar(50),
    scoll varchar (50),
    sadd varchar(50)
);
select * from student;
```

student.jsp

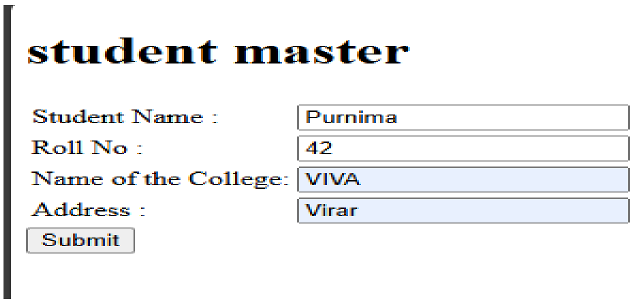
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>student master</h1>
<form action="response.jsp" method="post">
<table>
<tr><td>Student Name :</td>
<td><input type="text" name="text1" /></td></tr>
<tr><td>Roll No : </td>
<td><input type="text" name="text2" /></td></tr>
<tr><td>Name of the College:</td>
<td><input type="text" name="text3" /></td></tr>
<tr><td>Address :</td>
<td><input type="text" name="text4" /></td></tr>
</table>
<input type="submit" value="Submit" />
</form>
</body>
</html>
```

response.jsp

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

Programs based on web application development using JSP

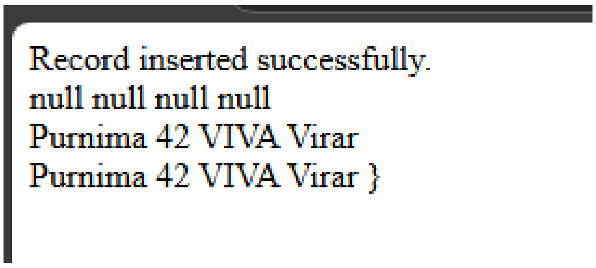
```
<!DOCTYPE html>
<html> <head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head> <body>
<%@page import="java.sql.*" %>
<%      try {
Class.forName("org.postgresql.Driver");
System.out.println("Drivers Registered");
Connection con=DriverManager.getConnection(
"jdbc:postgresql://localhost:5432/postgres","postgres","abc");
String t1, t2, t3, t4; t1=request.getParameter("text1");
t2=request.getParameter("text2");
t3=request.getParameter("text3");
t4=request.getParameter("text4");
PreparedStatement ps;
ps=con.prepareStatement("insert into student(sname,srollno,scoll,sadd) values(?,
?, ?, ?);");        ps.setString(1, t1);
ps.setString(2, t2); ps.setString(3, t3);
ps.setString(4, t4); ps.executeUpdate();
out.println("Record inserted successfully.");
Statement st=con.createStatement();
String sql="select * from student";
ResultSet rs=st.executeQuery(sql); while(rs.next())
{
String n1=rs.getString(1);
String n2=rs.getString(2);
String n3=rs.getString(3);
String n4=rs.getString(4);
%>             <br>
<% out.println(n1+" "+n2+" "+n3+" "+n4);}
} catch(Exception e){ out.println(e); } %>
</body>
</html>
```



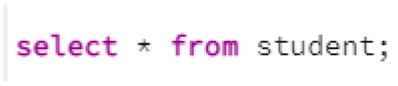
student master

Student Name :	Purnima
Roll No :	42
Name of the College:	VIVA
Address :	Virar

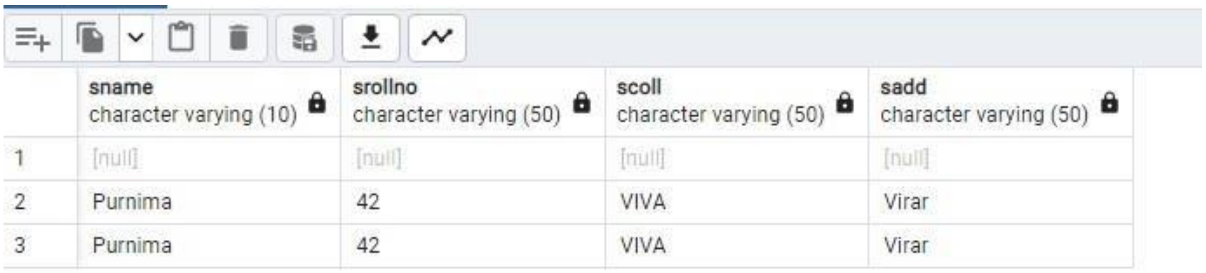
Submit



```
Record inserted successfully.
null null null null
Purnima 42 VIVA Virar
Purnima 42 VIVA Virar }
```



```
select * from student;
```



	sname character varying (10)	srollno character varying (50)	scoll character varying (50)	sadd character varying (50)
1	[null]	[null]	[null]	[null]
2	Purnima	42	VIVA	Virar
3	Purnima	42	VIVA	Virar

Programs based on web application development using JSP

Program no. 6

Aim: Delete Student Record AIM WAP to Delete records in

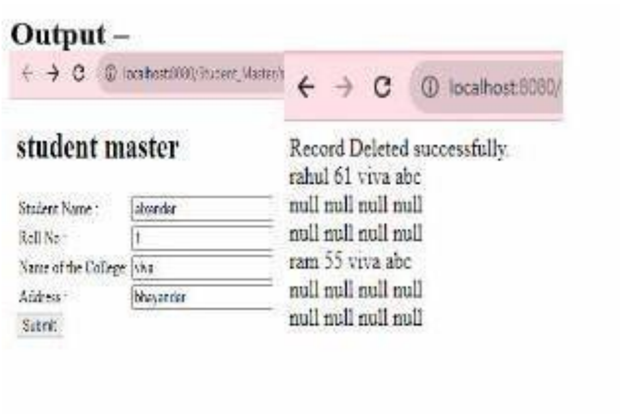
Student Master.

delete.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>
<h1>student master</h1>
<form action="response1.jsp" method="post">
<table>  <tr><td>Student Name :</td>
<td><input type="text" name="text1" /></td></tr>
<tr><td>Roll No : </td>
<td><input type="text" name="text2" /></td></tr>
<tr><td>Name of the College:</td>
<td><input type="text" name="text3" /></td></tr>
<tr><td>Address :</td>
<td><input type="text" name="text4" /></td></tr>
</table>
<input type="submit" value="Submit" />
</form> </body> </html>
```

response1.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html> <head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head> <body>
<%@page import="java.sql.*" %>
<%       try {
Class.forName("org.postgresql.Driver");
System.out.println("Drivers Registered");
Connection con=DriverManager.getConnection(
"jdbc:postgresql://localhost:5432/postgres","postgres","");
String t1, t2, t3, t4; t1=request.getParameter("text1");
t2=request.getParameter("text2"); t3=request.getParameter("text3");
t4=request.getParameter("text4");
PreparedStatement ps;
//insert into student(sname,srollno,scoll,sadd) values(?, ?, ?, ?);
ps=con.prepareStatement("DELETE FROM student where srollno= ?;");
ps.setString(1, t2); /*ps.setString(1, t1); ps.setString(3, t3);
ps.setString(4, t4);*/ ps.executeUpdate();
//out.println("Record inserted successfully.");
Statement st=con.createStatement();
String sql="select * from student";
ResultSet rs=st.executeQuery(sql); while(rs.next())
{ String n1=rs.getString(1);
 String n2=rs.getString(2);
 String n3=rs.getString(3);
 String n4=rs.getString(4);
 %>    <br>
 <% out.println(n1+" "+n2+" "+n3+" "+n4 ); }
} catch(Exception e){ out.println(e); } %>
</body> </html>
```



Output –

localhost:8080/Student_Master/

student master

Student Name : alsendar
Roll No : 1
Name of the College : viva
Address : bhayandar
Submit

localhost:8080/

```
Record Deleted successfully.
rahul 61 viva abc
null null null null
null null null null
ram 55 viva abc
null null null null
null null null null
```

Programs based on web application development using JSP

Program no. 7 Registration form

AIM Write a program to create registration form[JSP to Database pgADMIN

```
CREATE TABLE
        register( uid
        varchar(20), pass
        varchar(10),
        fname
        varchar(50),
        lname varchar
        (50),
        email varchar(50)
);
Select * from register
```

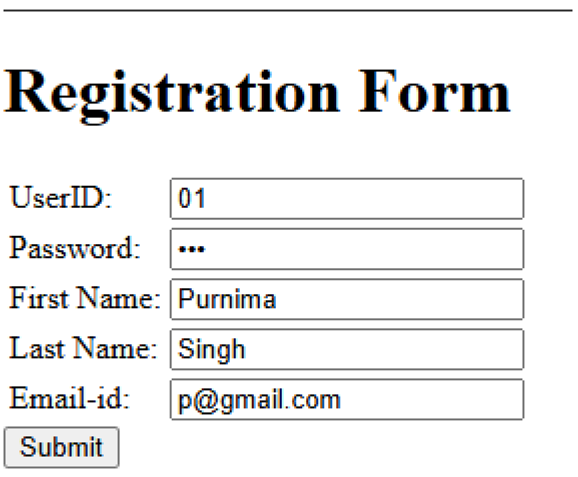
register.jsp

```
<%@ 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>
<h1>Registration Form</h1>
<form action="response.jsp" method="post">
<table>  <tr><td>UserID:</td>
<td><input type="text" name="text1" /></td></tr>
<tr><td>Password:</td>
<td><input type="password" name="text2" /></td></tr>
<tr><td>First Name:</td>
<td><input type="text" name="text3" /></td></tr>
<tr><td>Last Name:</td>
<td><input type="text" name="text4" /></td></tr>
<tr><td>Email-id:</td>
<td><input type="email" name="text5" /></td></tr>
</table>
<input type="submit" value="Submit" />
```

```
</form> </body> </html>
```

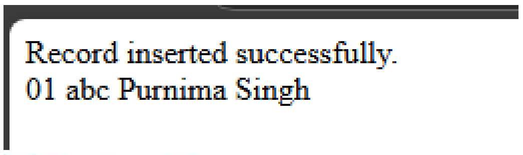
response.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html> <head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head> <body>
<%@page import="java.sql.*" %>
<%      try {
Class.forName("org.postgresql.Driver");
System.out.println("Registered");
Connection con=DriverManager.getConnection(
"jdbc:postgresql://localhost:5432/postgres","postgres","abc");
String t1, t2, t3, t4, t5; t1=request.getParameter("text1");
t2=request.getParameter("text2");
t3=request.getParameter("text3");
t4=request.getParameter("text4");
t5=request.getParameter("text5");
PreparedStatement ps;
ps=con.prepareStatement("insert into register(uid, pass, fname, lname, email)
values(?, ?, ?, ?, ?);");
ps.setString(1, t1);
ps.setString(2, t2); ps.setString(3, t3);
ps.setString(4, t4); ps.setString(5, t5);
ps.executeUpdate();
out.println("Record inserted successfully.");
Statement st=con.createStatement();
String sql="select * from register";
ResultSet rs=st.executeQuery(sql);
while(rs.next())            { String
n1=rs.getString(1);
String n2=rs.getString(2);
String n3=rs.getString(3);
String n4=rs.getString(4);
String n5=rs.getString(5);
%>
<br>
<% out.println(n1+" "+n2+" "+n3+" "+n4 ); }
} catch(Exception e){ out.println(e); } %>
</body> </html>
```

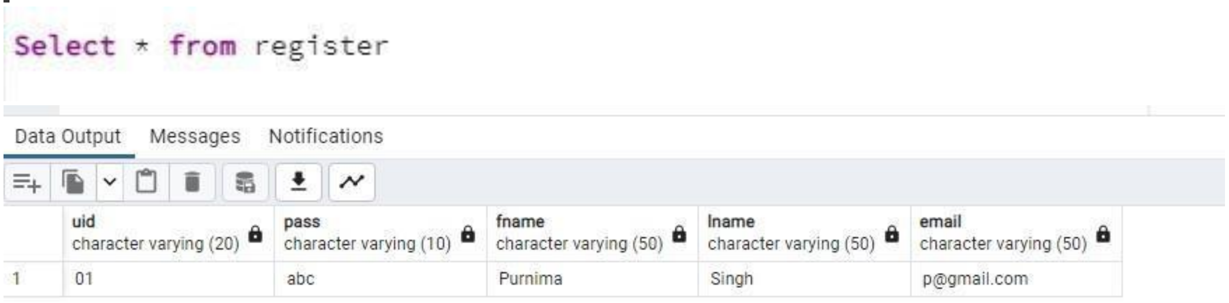


Registration Form

UserID:	01
Password:	•••
First Name:	Purnima
Last Name:	Singh
Email-id:	p@gmail.com



```
Record inserted successfully.
01 abc Purnima Singh
```



```
Select * from register
```

	uid character varying (20)	pass character varying (10)	fname character varying (50)	lname character varying (50)	email character varying (50)
1	01	abc	Purnima	Singh	p@gmail.com

Programs based on Assignment based Spring Framework

Practical no. 7

Program no. 1
AIM: Write a program to print “Hello World” using spring framework.

Steps to Create a Spring "Hello World" Application Step 1:

Create a New Maven Project

1. Open Eclipse.(2024)
2. Go to **File > New > Other...**.
3. In the wizard, select **Maven > Maven Project** and click **Next**.
4. Select **Create a simple project (skip archetype selection)** and click **Next**.
5. Fill in the **Group Id** (e.g., com.example) and **Artifact Id** (e.g., hello-spring) and click **Finish**.

Step 2: Update pom.xml

Open the pom.xmlfile in your project and add the Spring dependencies. Here’s a basic example:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>hello-spring</artifactId>
    <version>1.0-SNAPSHOT</version>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.10</version> <!-- Check for the latest version -->
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>1.8</source>
```

```
                        <target>1.8</target>
                    </configuration>
                </plugin>
            </plugins>
        </build>
</project>
```

Step 3: Create the Application Class

1. Right-click on the src/main/javadirectory, select **New > Package**, and name it com.example.hellospring.
2. Right-click on the newly created package, select **New > Class**, and name it HelloWorldApp.

Here's a simple example of what the class might look like:

```
package com.example.hellospring;

import org.springframework.context.ApplicationContext; import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class HelloWorldApp {
        public static void main(String[] args) { ApplicationContext
                context = new
AnnotationConfigApplicationContext(AppConfig.class);
                HelloWorld helloWorld = context.getBean(HelloWorld.class);
                helloWorld.sayHello();
        }
}
```

Step 4: Create the Configuration Class

1. Right-click on the same package, select **New > Class**, and name it AppConfig.

Here's an example of what the configuration class might look like:

```
package com.example.hellospring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class AppConfig { @Bean
        public HelloWorld helloWorld() { return
                new HelloWorld();
        }
}
```

Step 5: Create the HelloWorld Class

1. Right-click on the same package, select **New > Class**, and name it HelloWorld.

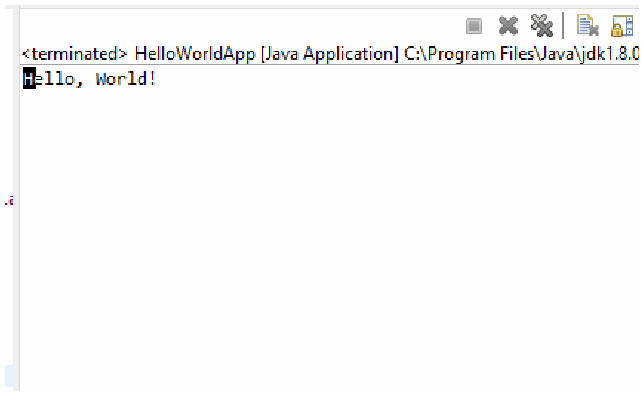
Here's how it might look:

```
packagecom.example.hellospring; public class

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

Step 6: Run the Application

1. Right-click on the HelloWorldApp.javafile and select **Run As > Java Application**.
2. You should see Hello, World!printed in the console.

Output:



```
<terminated> HelloWorldApp [Java Application] C:\Program Files\Java\jdk1.8.0
Hello, World!
```

Programs based on Assignment based Spring Framework

Program no. 2

AIM: Write a program to demonstrate dependency injection via setter method. Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>SpringDIExample</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <spring.version>5.3.22</spring.version>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-api</artifactId>
            <version>1.7.30</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>1.7.30</version>
        </dependency>
    </dependencies>
</project>
```

V8Engine.java

```
package di_program;

public class V8Engine implements Engine {
    @Override
    public void start() {
        System.out.println("V8 Engine is starting...");
    }
}
```

Programs based on Assignment based Spring Framework

Interface

```
package di_program;

public interface Engine {
	void start();
}
```

Car.java

```
package di_program;

public class Car {
	private Engine engine;

	// Setter method for dependency injection
	public void setEngine(Engine engine) {
		this.engine = engine;}
	public void startCar() {
		if (engine != null) {
			engine.start();
			System.out.println("Car is ready to go!");
		} else {
			System.out.println("Engine is not set. Cannot start the car.");
}}}
```

Appconfig.java

```
package di_program;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
@Configuration
public class AppConfig {
	@Bean
	public Engine engine() {
		return new V8Engine(); // Create and return the V8Engine bean
	}
	@Bean
	public Car car() {
		Car car = new Car();
		car.setEngine(engine()); // Inject the engine using the setter method
		return car;
	}}
```

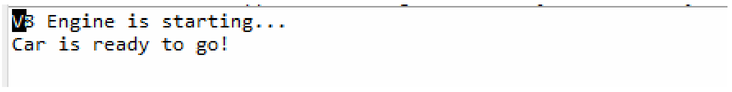
Main.java

```
package di_program;

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

Programs based on Assignment based Spring Framework

```
import org.springframework.context.annotation.AnnotationConfigApplicationContext;
public class Main {
	public static void main(String[] args) {
		// Create the application context from the configuration class
		ApplicationContext context = new
AnnotationConfigApplicationContext(AppConfig.class);

		// Retrieve the car bean
		Car car = context.getBean(Car.class);

		// Start the car
		car.startCar();
	}}
```

Output



```
V8 Engine is starting...
Car is ready to go!
```

Programs based on Assignment based Spring Framework

Program no. 3

AIM : Write a program to demonstrate dependency injection via Constructor.

Step 1: Create a New Maven Project

1. Open Eclipse.(from D drive)
2. Go to **File > New > Spring Legacy Project**
3. Give project name org.viva ,select simple java-> finish.
4. Expand Project->select src->create a package-> org.viva(if name is coming by default just click on finish).
5. Right click on package-> new->Class->Account.java
6. Again-> Right click on package-> new->Class->AccountTest.java
7. Select Src->Right Click->new->other->Bean Configuration File ->give name.
8. Select project-> right click->build path>configure build path->classpath-> add external jar->from d drive->open spring RELEASE->libs->select all jar files->apply->apply and close.
9. Run as JAVA Application-> Account.Test file.
10. Follow same steps for other program too.

Account.java

```
package org.viva;

public class Account {
	int acNo;
	String acName;
	double acbalance;
	/**
	  * @return the acNo
	  */
public int getAcNo() { return acNo;
	}
	/**
	  * @param acNo the acNo to set
	  */
	public void setAcNo(int acNo) {
		this.acNo = acNo;
	}
	/**
	  * @return the acName
	 */
	public String getAcName() {
		return acName;
	}
	/**
	 * @param acName the acName to set
	 */
	public void setAcName(String acName) {
		this.acName = acName;
	}
	/**
	 * @return the acbalance
	 */
	public double getAcbalance() {
		return acbalance;
	}
	/**
	 * @param acbalance the acbalance to set
	 */
	public void setAcbalance(double acbalance) {


 this.acbalance = acbalance;
	}
	public Account(int acNo, String acName, double acbalance) {
		super();
		this.acNo = acNo;
		this.acName = acName;
		this.acbalance = acbalance;
	}

	public Account() {
		super();
	}
 }
```

Appctx.xml

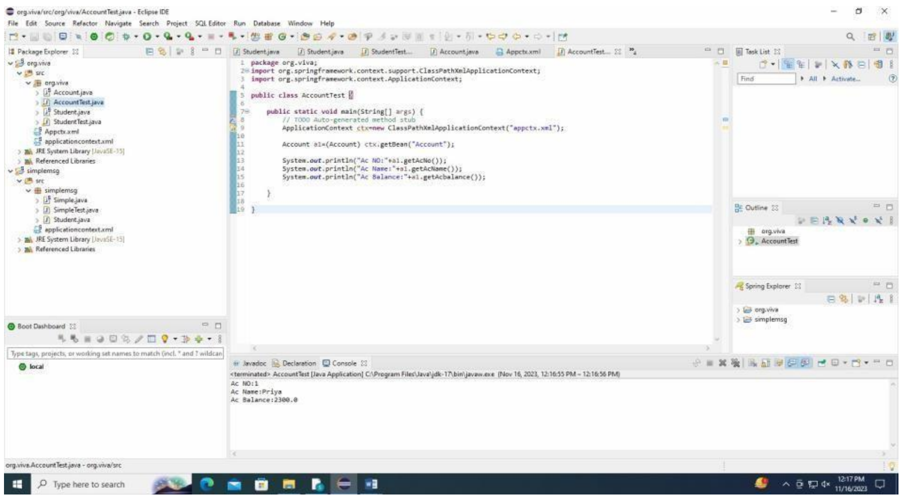
```
 <?xml version="1.0" encoding="UTF-8"?>
 <beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://www.springframework.org/schema/beans
 http://www.springframework.org/schema/beans/spring-beans.xsd">

 <bean id="Account" class="org.viva.Account">
	<constructor-arg type="int" value="000001" >
```

```
                        </constructor-arg>
          <constructor-arg type="String" value="Priya">
                    </constructor-arg>
               <constructor-arg type="double" value="2300">
</constructor-arg>
</bean>
</beans>
```

AcoountTest.java

```
package org.viva;
import
org.springframework.context.support.ClassPathXmlApplicationC
ontext; import org.springframework.context.ApplicationContext;
public class AccountTest {
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                ApplicationContext ctx=new

                ClassPathXmlApplicationContext("appctx.xml"); Account

                a1=(Account) ctx.getBean("Account");

                System.out.println("Ac
                NO:"+a1.getAcNo());
                System.out.println("Ac
                Name:"+a1.getAcName());
                System.out.println("Ac
                Balance:"+a1.getAcbalance());}}
```

OUTPUT:



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

public class AccountTest {

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

        Account a1=(Account) ctx.getBean("Account");

        System.out.println("Ac NO:"+a1.getAcNo());
        System.out.println("Ac Name:"+a1.getAcName());
        System.out.println("Ac Balance:"+a1.getAcbalance());

    }

}
```

```
<terminated> AccountTest [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe (Nov 16, 2023, 12:16:55 PM – 12:16:56 PM)
Ac NO:1
Ac Name:Priya
Ac Balance:2300.0
```

Programs based on Assignment based Spring Framework

Program no. 4

AIM : Write a program to demonstrate Autowiring.

Engine.java

```
package myspring.viva;

import org.springframework.stereotype.Component;

@Component
public class Engine {
	public void start() {
		System.out.println("Engine started!");
	}
}
```

Car.java

```
package myspring.viva;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class Car {
	private Engine engine;

	// Autowire the Engine class into the Car class @Autowired
	public Car(Engine engine) {
		this.engine = engine;
	}

	public void drive() {
		engine.start();
		System.out.println("Car is moving!");
	}
}
```

Spring-config.xml

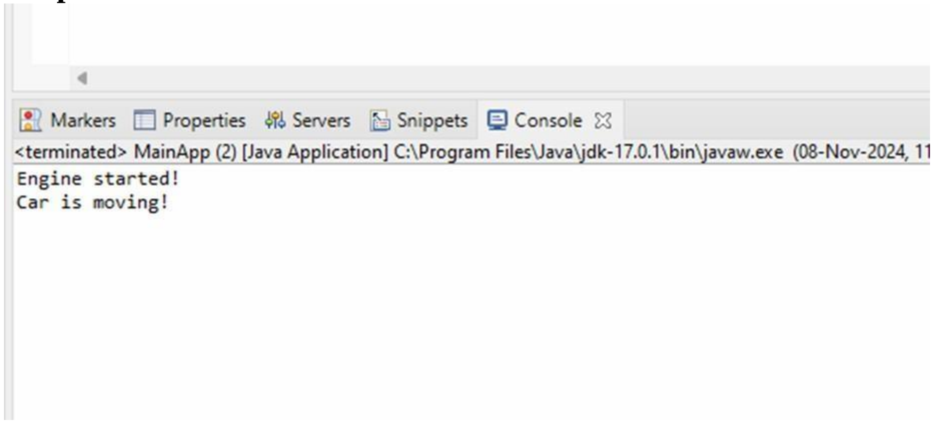
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns:context="http://www.springframework.org/schema/context"
	xsi:schemaLocation="
		http://www.springframework.org/schema/beans
		http://www.springframework.org/schema/beans/spring-beans.xsd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

```
        <!-- Enable component scanning for the 'myspring.viva' package -->
        <context:component-scan base-package="myspring.viva" />
</beans>
```

MainApp.java

```
package myspring.viva;

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

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

                // Get the Car bean from the Spring container Car
                car1 = context.getBean(Car.class);
                car1.drive();

                ((ClassPathXmlApplicationContext) context).close();
        }
}
```

Output:



```
<terminated> MainApp (2) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (08-Nov-2024, 11
Engine started!
Car is moving!
```

PROGRAMS BASED ON AOP CONCEPTS

Practical No.:08

Program no. 1
Aim: Write a program to demonstrate Spring AOP – before advice..
Logging.java
```
package org.mca1;
public class Logging {
	public void beforeAdvice() {
		System.out.println("Going to setup student profile.");
	}
}
```

Student.java
```
package org.mca1;
public class Student {
	int age;
	String name;

	public int getAge() {
		return age;
	}

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

	public String getName() {
		return name;
	}

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

StudentTest.java
```
package org.mca1;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.io.*;
public class StudentTest {
    private static ApplicationContext context;
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		 context = new ClassPathXmlApplicationContext("appctx.xml");
		Student student = (Student) context.getBean("student");
		String abc=student.getName();
```

```
                System.out.println("Name is:"+abc);
                int ag=student.getAge();
                System.out.println("Age is:"+ag);
        }

}
```

In src/main/resources
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"
        xmlns:aop="http://www.springframework.org/schema/aop"
        xmlns:context="http://www.springframework.org/schema/context"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
                http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">

<aop:config>
    <aop:aspect id = "log" ref = "logging">
      <aop:pointcut id = "selectAll"
        expression = "execution(* org.mca1.Student.getName(..))"/>

      <aop:before pointcut-ref="selectAll" method="beforeAdvice"/>
    </aop:aspect>
  </aop:config>
  <bean id = "student" class = "org.mca1.Student">
    <property name = "name"  value = "Zara" />
    <property name = "age"  value = "11"/>
  </bean>

  <!-- Definition for logging aspect -->
  <bean id = "logging" class = "org.mca1.Logging"/>
</beans>
```

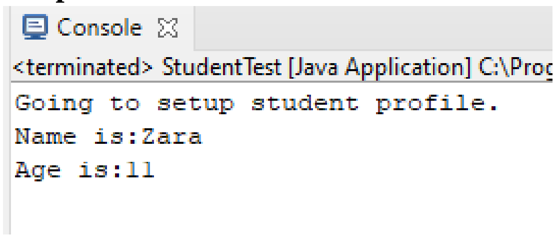
Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.mca1</groupId>
```

```
 <artifactId>org.mca1</artifactId>
 <version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

 <name>AopStudent</name>
 <url>http://maven.apache.org</url>

 <properties>
   <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 </properties>

 <dependencies>
   <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>3.8.1</version>
     <scope>test</scope>
   </dependency>

   <dependency>
           <groupId>org.springframework</groupId>
           <artifactId>spring-core</artifactId>
           <version>5.2.8.RELEASE</version>
       </dependency>

       <!--  https://mvnrepository.com/artifact/org.springframework/spring-context  -->
       <dependency>
           <groupId>org.springframework</groupId>
           <artifactId>spring-context</artifactId>
           <version>5.2.8.RELEASE</version>
       </dependency>
       <!--  https://mvnrepository.com/artifact/org.postgresql/postgresql  -->
       <dependency>
           <groupId>org.postgresql</groupId>
           <artifactId>postgresql</artifactId>
           <version>42.2.19</version>
       </dependency>

       <!--  https://mvnrepository.com/artifact/org.springframework/spring-jdbc  -->
       <dependency>
           <groupId>org.springframework</groupId>
           <artifactId>spring-jdbc</artifactId>
           <version>5.2.8.RELEASE</version>
       </dependency>
       <!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
       <dependency>
```

```
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>1.9.6</version>
        </dependency>
        <!--  https://mvnrepository.com/artifact/org.aspectj/aspectjweaver  -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.9.6</version>
        </dependency>
  </dependencies>
</project>
```

Output:



```
<terminated> StudentTest [Java Application] C:\Prog
Going to setup student profile.
Name is:Zara
Age is:11
```

PROGRAMS BASED ON AOP CONCEPTS

Program no.2
Aim: Write a program to demonstrate Spring AOP – after advice.
Logging.java

```
package org.mca2;
public class Logging {
	/**
	  * This is the method which I would like to execute
	  * after a selected method execution.
	  */
	  public void afterAdvice(){
	    System.out.println("Student profile setup complete.");
	  }

}
```

Student.java

```
package org.mca2;
public class Student {
	int age;
	String name;

	public int getAge() {
		return age;
	}

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

	public String getName() {
		return name;
	}

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

PROGRAMS BASED ON AOP CONCEPTS

StudentTest.java

```
package org.mca2;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.io.*;
public class StudentTest {
    private static ApplicationContext context;
            public static void main(String[] args) {
                        // TODO Auto-generated method stub
                         context = new ClassPathXmlApplicationContext("appctx.xml");
                        Student student = (Student) context.getBean("student");
                        String abc=student.getName();
                        System.out.println("Name is:"+abc);
                        int ag=student.getAge();
                        System.out.println("Age is:"+ag);
            }

}
```

In src/main/resources
appctx.xml

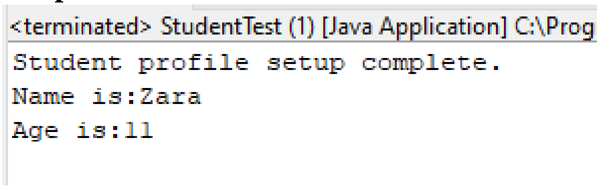
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:aop="http://www.springframework.org/schema/aop"
            xmlns:context="http://www.springframework.org/schema/context"
            xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                        http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.3.xsd
                        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">

<aop:config>
      <aop:aspect id = "log" ref = "logging">
         <aop:pointcut id = "selectAll"
            expression = "execution(* org.mca2.Student.getName(..))"/>

         <aop:after pointcut-ref="selectAll" method="afterAdvice"/>
      </aop:aspect>
   </aop:config>
   <bean id = "student" class = "org.mca2.Student">
      <property name = "name"  value = "Zara" />
      <property name = "age"  value = "11"/>
   </bean>

   <!-- Definition for logging aspect -->
   <bean id = "logging" class = "org.mca2.Logging"/>
</beans>
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.mca2</groupId>
  <artifactId>org.mca2</artifactId>
  <version>0.0.1-SNAPSHOT</version>
   <packaging>jar</packaging>

  <name>AopStudent</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>

        <!--  https://mvnrepository.com/artifact/org.springframework/spring-context  -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>
        <!--  https://mvnrepository.com/artifact/org.postgresql/postgresql  -->
        <dependency>
```

```
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.2.19</version>
        </dependency>

        <!--  https://mvnrepository.com/artifact/org.springframework/spring-jdbc  -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>1.9.6</version>
        </dependency>
        <!--  https://mvnrepository.com/artifact/org.aspectj/aspectjweaver  -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.9.6</version>
        </dependency>

  </dependencies>

</project>
```

Output:



```
<terminated> StudentTest (1) [Java Application] C:\Prog
Student profile setup complete.
Name is:Zara
Age is:11
```

PROGRAMS BASED ON AOP CONCEPTS

Program no. 3
Aim: Write a program to demonstrate Spring AOP – after returning advice.

Student.java:
```
package org.mca;
public class Student {
        private Integer age;
        private String name;
        public void setAge(Integer age) {
          this.age = age;
        }
        public Integer getAge() {
          System.out.println("Age : " + age );
          return age;

        }
        public void setName(String name) {
          this.name = name;
        }
        public String getName() {
          System.out.println("Name : " + name );
          return name;
        }
        public void printThrowException(){
          System.out.println("Exception raised");
          throw new IllegalArgumentException();
        }
      }
```

Logging.java:
```
package org.mca;

public class Logging {
       public void afterReturningAdvice(Object retVal){
          System.out.println("Returning:" + retVal.toString() );
        }

}
```

Beans.xml:
```
<?xml version = "1.0" encoding = "UTF-8"?>
<beans xmlns = "http://www.springframework.org/schema/beans"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xmlns:aop = "http://www.springframework.org/schema/aop"
  xsi:schemaLocation = "http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
  http://www.springframework.org/schema/aop
  http://www.springframework.org/schema/aop/spring-aop-3.0.xsd ">
```

```
   <aop:config>
      <aop:aspect id = "log" ref = "logging">
         <aop:pointcut id = "selectAll"
         expression = "execution(* org.mca.*.*(..))"/>
         <aop:after-returning pointcut-ref = "selectAll"
            method = "afterReturningAdvice" returning = "retVal"/>
      </aop:aspect>
   </aop:config>

   <!-- Definition for student bean -->
   <bean id = "student" class = " org.mca.Student">
      <property name = "name"  value = "Zara" />
      <property name = "age"  value = "11"/>
   </bean>

   <!-- Definition for logging aspect -->
   <bean id = "logging" class = " org.mca.Logging"/>
</beans>
```

MainApp.java:

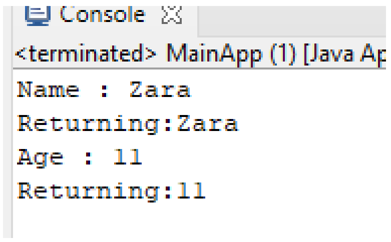
```
package org.mca;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                ApplicationContext context = new
ClassPathXmlApplicationContext("Beans.xml");

            Student student = (Student) context.getBean("student");
            student.getName();
            student.getAge();
          }

        }
```

PROGRAMS BASED ON AOP CONCEPTS

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.tutorialspoint</groupId>
  <artifactId>AfterReturningAdvise</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>AopStudent</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.2.19</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>1.9.6</version>
        </dependency>
        <!--  https://mvnrepository.com/artifact/org.aspectj/aspectjweaver  -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.9.6</version>
        </dependency>


  </dependencies>
</project>
```

Output



```
Name : Zara
Returning:Zara
Age : 11
Returning:11
```

PROGRAMS BASED ON AOP CONCEPTS

Program no 4:

Aim: Write a program to demonstrate Spring AOP –around advice.

Student.java

```
package org.viva2;

public class Student {
        private Integer age;
          private String name;
          public void setAge(Integer age) {
            this.age = age;
          }
          public Integer getAge() {
            System.out.println("Age : " + age );
            return age;
          }
          public void setName(String name) {
            this.name = name;
          }
          public String getName() {
            System.out.println("Name : " + name );
            return name;
          }
          public void printThrowException(){
            System.out.println("Exception raised");
            throw new IllegalArgumentException();
          }
}
```

Logging.java

```
package org.viva2;
import org.aspectj.lang.ProceedingJoinPoint;
public class Logging {
         /**
    * This is the method which I would like to execute
    * around a selected method execution.
  */
  public String aroundAdvice(ProceedingJoinPoint jp) throws Throwable{
    System.out.println("Around advice");
    Object[] args = jp.getArgs();
    if(args.length>0){
      System.out.print("Arguments passed: " );
      for (int i = 0; i < args.length; i++) {
        System.out.print("arg "+(i+1)+": "+args[i]);
      }
    }
    Object result = jp.proceed(args);
    System.out.println("Returning " + result);
    return result.toString();
  }
}
```

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

   <aop:config>
      <aop:aspect id = "log" ref = "logging">
         <aop:pointcut id = "selectName"
            expression = "execution(* org.viva2.Student.getName(..))"/>
         <aop:around pointcut-ref = "selectName" method = "aroundAdvice"/>
      </aop:aspect>
   </aop:config>

   <!-- Definition for student bean -->
    <bean id = "student" class = "org.viva2.Student">
       <property name = "name"  value = "Sanjana" />
      <property name = "age"  value = "21"/>
   </bean>

   <!-- Definition for logging aspect -->
   <bean id = "logging" class = "org.viva2.Logging"/>
</beans>
```

StudentTest.java

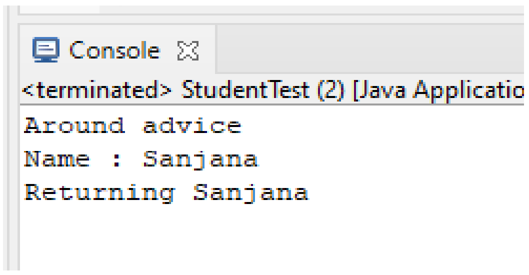
```
package org.viva2;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class StudentTest {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		 ApplicationContext context = new
ClassPathXmlApplicationContext("Appctx.xml");

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

```
                student.getName();
        }}
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.viva2</groupId>
  <artifactId>org.viva2</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>AopStudent</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.2.19</version>
        </dependency>

        <!--  https://mvnrepository.com/artifact/org.springframework/spring-jdbc  -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>1.9.6</version>
        </dependency>
        <!--  https://mvnrepository.com/artifact/org.aspectj/aspectjweaver  -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.9.6</version>
        </dependency>



  </dependencies>

</project>
```

Output


Console

```
<terminated> StudentTest (2) [Java Applicatio
Around advice
Name : Sanjana
Returning Sanjana
```

PROGRAMS BASED ON AOP CONCEPTS

Program no. 5: Write a program to demonstrate Spring AOP –After Throwing advice.

Student.java:

```
package org.viva4;

public class Student {
             private Integer age;
               private String name;
            public Integer getAge() {
                        return age;
            }
            public void setAge(Integer age) {
                        this.age = age;
            }
            public String getName() {
                        return name;
            }
            public void setName(String name) {
                        this.name = name;
            }
             public void printThrowException(){
                  System.out.println("Exception raised");
                  throw new IllegalArgumentException();
               }
}
```

Logging.java:

```
package org.viva4;

import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.AfterThrowing;
@Aspect

public class Logging {

   public void afterThrowingAdvice(JoinPoint jp, Throwable error){
      System.out.println("Method Signature: "  + jp.getSignature());
      System.out.println("Exception: "+error);
   }
}
```

Beans.xml:

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

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

  <aop:aspectj-autoproxy/>

  <!-- Definition for student bean -->
  <bean id = "student" class = " org.viva4.Student">
    <property name = "name"  value = "Zara" />
    <property name = "age"  value = "11"/>
  </bean>

  <!-- Definition for logging aspect -->
  <bean id = "logging" class = " org.viva4.Logging"/>

</beans>
```

MainApp.java:

```
package org.viva4;

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

public class MainApp {


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

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

          }
```

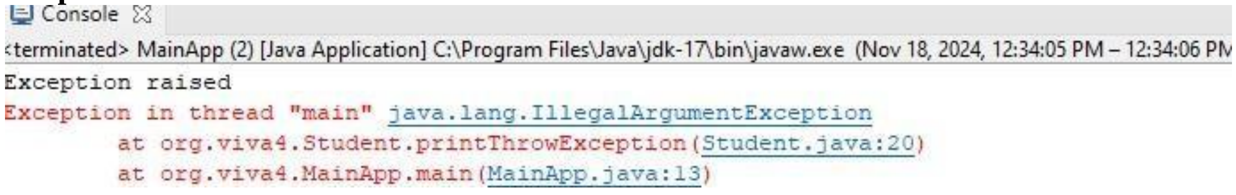
Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.tutorialspoint</groupId>
  <artifactId>AfterThrowing</artifactId>
  <version>0.0.1-SNAPSHOT</version>
   <packaging>jar</packaging>
```

PROGRAMS BASED ON AOP CONCEPTS

```
<name>AopStudent</name>
<url>http://maven.apache.org</url>

<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>

  <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>

        <!--  https://mvnrepository.com/artifact/org.springframework/spring-context  -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>
        <!--  https://mvnrepository.com/artifact/org.postgresql/postgresql  -->
        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <version>42.2.19</version>
        </dependency>

        <!--  https://mvnrepository.com/artifact/org.springframework/spring-jdbc  -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.2.8.RELEASE</version>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.aspectj/aspectjrt -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>1.9.6</version>
        </dependency>
        <!--  https://mvnrepository.com/artifact/org.aspectj/aspectjweaver  -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.9.6</version>
        </dependency>



  </dependencies>
</project>
```

Output:



```
Console
<terminated> MainApp (2) [Java Application] C:\Program Files\Java\jdk-17\bin\javaw.exe  (Nov 18, 2024, 12:34:05 PM – 12:34:06 PM
Exception raised
Exception in thread "main" java.lang.IllegalArgumentException
	at org.viva4.Student.printThrowException(Student.java:20)
	at org.viva4.MainApp.main(MainApp.java:13)
```

PROGRAMS BASED ON AOP CONCEPTS

Program 6:
Aim: Write a program to demonstrate Spring AOP – pointcuts.

MyService.java

```java
package com.example.aopdemo;

public class MyService {
    public void performTask() {
        System.out.println("Executing the task in MyService.");
    }
}
```

LoggingAspect.java

```java
package com.example.aopdemo;

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

@Aspect
public class LoggingAspect {

    // Define a pointcut expression
    @Before("execution(* com.example.aopdemo.MyService.performTask(..))")
    public void logBefore() {
        System.out.println("LoggingAspect: Before executing performTask.");
    }
}
```

applicationContext.xml

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

    <!-- Enable AspectJ auto proxy -->
    <aop:aspectj-autoproxy/>

    <!-- Register beans -->
    <bean id="myService" class="com.example.aopdemo.MyService"/>
    <bean id="loggingAspect" class="com.example.aopdemo.LoggingAspect"/>

</beans>
```

PROGRAMS BASED ON AOP CONCEPTS

MainApp.java

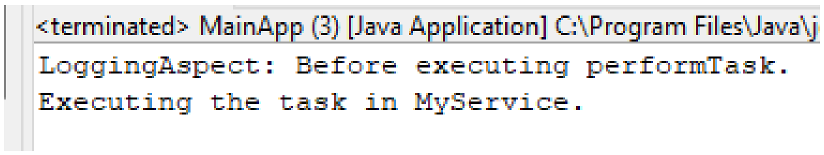
```
package com.example.aopdemo;

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

public class MainApp {
   public static void main(String[] args) {
      // Load the Spring context
      ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

      // Retrieve the service bean
      MyService myService = context.getBean("myService", MyService.class);

      // Call the method to see AOP in action
      myService.performTask();

      // Close the context
      ((ClassPathXmlApplicationContext) context).close();
   }
}
```

Pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.example</groupId>
   <artifactId>SpringAOPDemo</artifactId>
   <version>1.0-SNAPSHOT</version>

   <dependencies>
      <!-- Spring Context for AOP -->
      <dependency>
         <groupId>org.springframework</groupId>
         <artifactId>spring-context</artifactId>
         <version>5.3.30</version>
      </dependency>

      <!-- Spring AOP -->
      <dependency>
         <groupId>org.springframework</groupId>
         <artifactId>spring-aop</artifactId>
```

```
            <version>5.3.30</version>
        </dependency>

        <!-- AspectJ for annotations -->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.9.19</version>
         </dependency>
    </dependencies>
</project>
```

Output:



```
<terminated> MainApp (3) [Java Application] C:\Program Files\Java\j
LoggingAspect: Before executing performTask.
Executing the task in MyService.
```

Programs Based on Spring JDBC

Practical No 9

Program no. 1
Aim: Write a program to demonstrate Spring JdbcTemplate class to store data in database table.

Program:

On pgAdmin4

```
create table mymovies(mid int,title varchar,actor varchar);

select * from mymovies;
```

Movies.java

```
package org.viva;

public class Movies {

	int mid;
	String title;
	String actor;
	public Movies(int mid, String title, String actor) {
		super();
		this.mid = mid;
		this.title = title;
		this.actor = actor;
	}
	/**
	 * @return the mid
	 */
	public int getMid() {
		return mid;
	}
	/**
	 * @param mid the mid to set
	 */
	public void setMid(int mid) {
		this.mid = mid;
	}
	/**
	 * @return the title
	 */
	public String getTitle() {
		return title;
	}
	/**
	 * @param title the title to set
	 */
	public void setTitle(String title) {
		this.title = title;
	}
	/**
	 * @return the actor
	 */
	public String getActor() {
		return actor;
	}
	/**
	 * @param actor the actor to set
	 */
	public void setActor(String actor) {
		this.actor = actor;
	}



}
```

MoviesDao.java

```
package org.viva;
import org.springframework.jdbc.core.JdbcTemplate;

public class MoviesDao {

    private JdbcTemplate jdbcTemplate;

    // Getter method for jdbcTemplate
    public JdbcTemplate getJdbcTemplate() {
        return jdbcTemplate;
    }

    // Setter method for jdbcTemplate
    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    // Method to insert movie record into the database
    public int insMovie(Movies m1) {
        // Use parameterized SQL to avoid SQL injection
        String insSql = "INSERT INTO mymovies (mid, title, actor) VALUES (?, ?, ?)";
        // Use jdbcTemplate's update method with parameters to safely insert data
        return jdbcTemplate.update(insSql, m1.getMid(), m1.getTitle(), m1.getActor());
    }

}
```

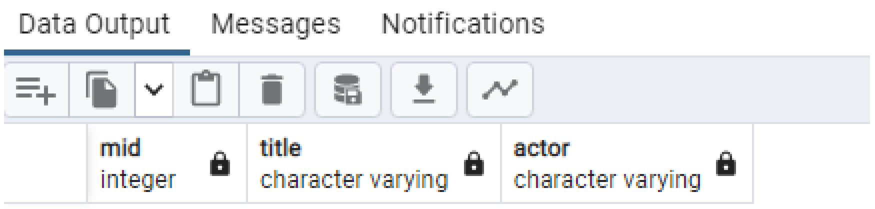
MoviesTest.java

```
package org.viva;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MoviesTest {

	public static void main(String[] args) {

		// TODO Auto-generated method stub



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

		MoviesDao m1=(MoviesDao)appCon.getBean("moviebean");

		Movies t1=new Movies(1,"A Beautiful Mind","Russel Crow");

		System.out.println(m1.insMovie(t1));

	}

}
```

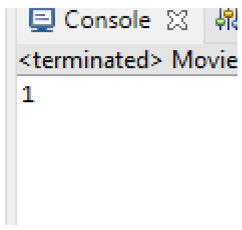
Appctx.xml

```
<?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:5433/postgres" />
<property name="username" value="postgres" />
<property name="password" value="abc" />
</bean>

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

<bean id="moviebean" class="org.viva.MoviesDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>

</beans>
```



mid integer	title character varying	actor character varying



```
<terminated> Movie
1
```

Programs Based on Spring JDBC

Program no. 2
Aim: Write a program to demonstrate Spring JdbcTemplate class to store data in database table Employee and also demonstrate update and delete.

Program:

On pgAdmin

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

select * from employee;
```

Employee.java

```
package org.viva;

public class Employee {

	private int id;
	private String name;
	private int salary;

	public Employee() {
		super();
		// TODO Auto-generated constructor stub
	}

	public int getId() {
		return id;
	}



	public void setId(int id) {
		this.id = id;
	}

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

EmployeeDao.java

```
package org.viva;
import org.springframework.jdbc.core.JdbcTemplate;

public class EmployeeDao {

	private JdbcTemplate jdbcTemplate;

	public EmployeeDao() {
		super();
		// TODO Auto-generated constructor stub
	}

	public EmployeeDao(JdbcTemplate jdbcTemplate) {
		super();
		this.jdbcTemplate = jdbcTemplate;
	}

	public JdbcTemplate getJdbcTemplate() {
		return jdbcTemplate;
	}

	public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
		this.jdbcTemplate = jdbcTemplate;
	}

	public int saveEmployee(Employee e){
		 String query="insert into Employee
values('"+e.getId()+"','"+e.getName()+"','"+e.getSalary()+"')";
		 return jdbcTemplate.update(query);		}


	public int updateEmployee(Employee e){
		 String query="update employee set
name='"+e.getName()+"',salary='"+e.getSalary()+"' where id='"+e.getId()+"' ";
```

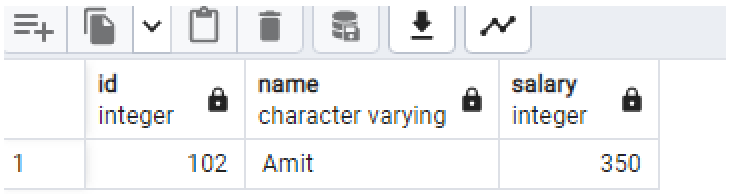
```
                 return jdbcTemplate.update(query);
                }
                public int deleteEmployee(Employee e){
                 String query="delete from employee where id='"+e.getId()+"' ";
                 return jdbcTemplate.update(query);
                }

        }
```

appctx1.xml

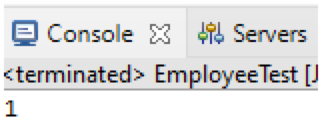
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">



<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="abc" />
</bean>

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


<bean id="edao" class="org.viva.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>

</beans>
```

EmployeeTest.java

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

public class EmployeeTest {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                ApplicationContext ctx=new ClassPathXmlApplicationContext("appctx1.xml");
                EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
                int status=dao.saveEmployee(new Employee(102,"Amit",350));
                System.out.println(status);

                /*int status=dao.updateEmployee(new
Employee(102,"Sonoo",15000));
                System.out.println(status);
                */

                /*Employee e=new Employee();
                e.setId(102);
                int status=dao.deleteEmployee(e);
                System.out.println(status);*/
    }

}
```



	id integer	name character varying	salary integer
1	102	Amit	350



```
Console  Servers
<terminated> EmployeeTest [J
1
```

Programs Based on Spring JDBC

Program no. 3
Aim: Write a program to demonstrate RowMapper interface to fetch the records from the database.

Program:

On pgAdmin

```
create table emp1(id int,name varchar);

select *from emp1;
```

Employee.java

```
package com.viva;
public class Employee {
	int id;
	String name;
	public Employee() {
		super();
	}
	public Employee(int id, String name) {
		super();
		this.id = id;
		this.name = name;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}

}
```

Programs Based on Spring JDBC

EmployeeDao.java

```
package com.viva;
import org.springframework.jdbc.core.*;
import java.util.*;
public class EmployeeDao {

        JdbcTemplate jdbcTemplate;

        public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
                this.jdbcTemplate = jdbcTemplate;
        }
        public int saveEmp(Employee e){
           String query="insert into emp1 values("+e.getId()+",'"+e.getName()+"')";
           return jdbcTemplate.update(query);
        }
        public List<Employee> findAll() {

           String sql = "SELECT * FROM emp1";

           List<Employee> obj = jdbcTemplate.query(sql,new EmpRowMapper());

           return obj;

        }
        public int saveEmp1(Employee e1) {
                // TODO Auto-generated method stub
                return 0;
        }
}
```

EmployeeTest.java

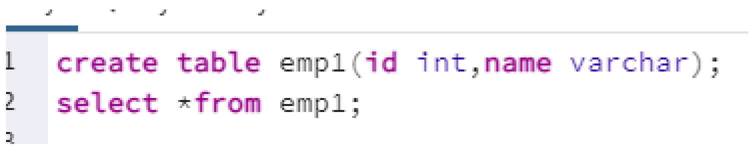
```
package com.viva;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.util.*;
public class EmployeeTest {
        private static ApplicationContext appCon;
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                appCon = new ClassPathXmlApplicationContext("appctx.xml");
                EmployeeDao fac=(EmployeeDao)appCon.getBean("Emp1");
                Employee e1=new Employee(2,"Sonia");
                System.out.println(fac.saveEmp(e1));
                List<Employee> lstemp=fac.findAll();
                for(Employee e2:lstemp)
                {
                        System.out.print(e2.getId());
                        System.out.println(e2.getName());
                }}}
```

EmpRowMapper.java

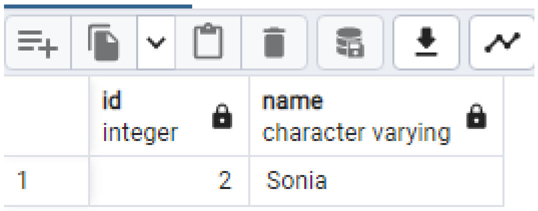
```
package com.viva;
import org.springframework.jdbc.core.RowMapper;
import java.sql.ResultSet;
import java.sql.SQLException;
public class EmpRowMapper implements RowMapper<Employee>
{
        @Override
        public Employee mapRow(ResultSet arg0, int arg1) throws SQLException
        {
                Employee e1=new Employee();
                e1.setId(arg0.getInt(1));
                e1.setName(arg0.getString(2));
                return e1;
        }}
```

Appctx.xml

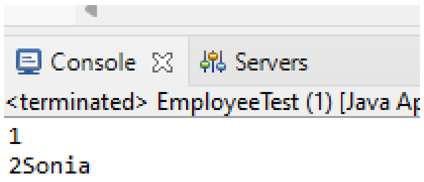
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
<bean 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="abc" />
</bean>
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
<property name="dataSource" ref="ds"></property>
</bean>
<bean id="Emp1" class="com.viva.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>
</beans>
```



```
create table emp1(id int,name varchar);
select *from emp1;
```



	id integer	name character varying
1	2	Sonia



```
<terminated> EmployeeTest (1) [Java Ap
1
2Sonia
```

Programs Based on Spring JDBC

Program no. 4
Aim: Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface

On pgAdmin

```
create table employee23(id int,name varchar, salary int );

select * from employee23;

insert into employee23 values(102,'sonia');

insert into employee23 values(102,'sonu');

select * from employee23;
```

Employee.java

```
package org.viva23;

public class Employee {

	private int id;
	private String name;
	private int salary;



	public Employee() {
		super();
		// TODO Auto-generated constructor stub
	}



	public int getId() {
		return id;
	}



	public void setId(int id) {
		this.id = id;
	}



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



	public int getSalary() {
		return salary;
	}



	public void setSalary(int salary) {
		this.salary = salary;
	}



	public Employee(int id, String name, int salary) {
		super();
		this.id = id;
		this.name = name;
		this.salary = salary;
	}



	@Override
	public String toString() {
		return "Employee [id=" + id + ", name=" + name + ", salary=" + salary +
"]";
	}




}
```

EmployeeDao.java

```
package org.viva23;
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 EmployeeDao {
	private JdbcTemplate jdbcTemplate;

	public EmployeeDao() {
		super();
		// TODO Auto-generated constructor stub
	}

	public EmployeeDao(JdbcTemplate jdbcTemplate) {
		super();
		this.jdbcTemplate = jdbcTemplate;
	}

	public JdbcTemplate getJdbcTemplate() {
		return jdbcTemplate;
	}

	public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
		this.jdbcTemplate = jdbcTemplate;
	}

	public List<Employee> getAllEmployees(){
		 return jdbcTemplate.query("select * from employee23",new
ResultSetExtractor<List<Employee>>(){
				 @Override
				  public List<Employee> extractData(ResultSet rs) throws
SQLException,
						 DataAccessException {

					List<Employee> list=new ArrayList<Employee>();
					while(rs.next()){
					Employee e=new Employee();
					e.setId(rs.getInt(1));
					e.setName(rs.getString(2));
					e.setSalary(rs.getInt(3));
					list.add(e);
					}
					return list;
					}
				 });
				}

}
```

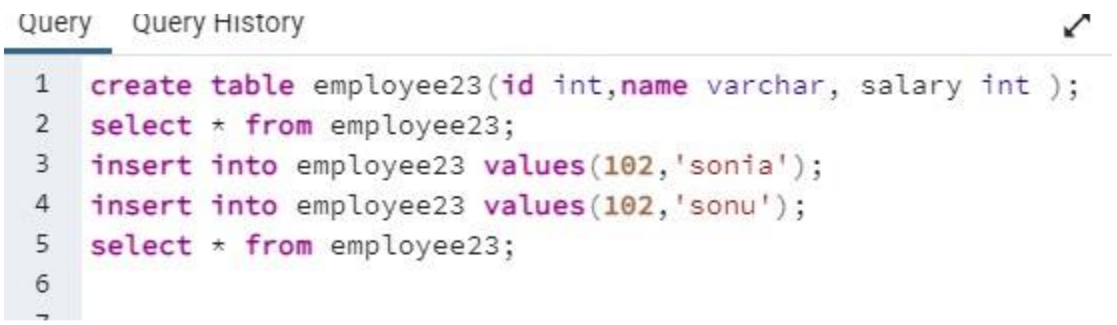
Appctx.xml

```
<?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">
```

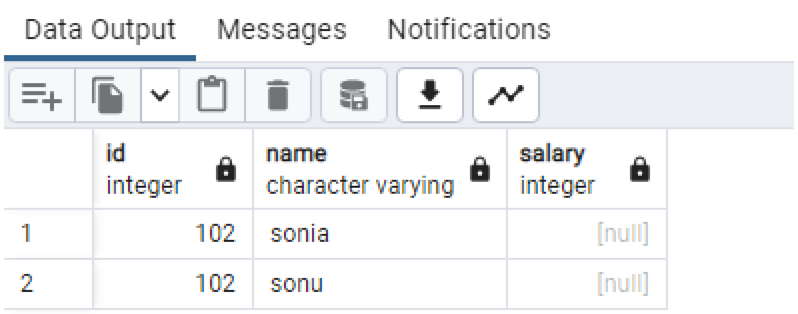
```
<bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName" value="org.postgresql.Driver" />
<property name="url" value="jdbc:postgresql://localhost:5432/postgres" />
<property name="username" value="postgres" />
<property name="password" value="abc"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

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

<bean id="edao" class="org.viva23.EmployeeDao">
<property name="jdbcTemplate" ref="jdbcTemplate"></property>
</bean>

</beans>
```

EmployeeTest.java

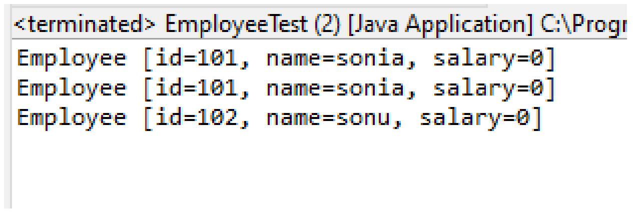
```
package org.viva23;

import java.util.List;

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

public class EmployeeTest {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		ApplicationContext ctx=new 
ClassPathXmlApplicationContext("appctx.xml");
           EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
           List<Employee> list=dao.getAllEmployees();

           for(Employee e:list)
               System.out.println(e);


	}

}
```



```
create table employee23(id int,name varchar, salary int );
select * from employee23;
insert into employee23 values(102,'sonia');
insert into employee23 values(102,'sonu');
select * from employee23;
```



	id integer	name character varying	salary integer
1	102	sonia	[null]
2	102	sonu	[null]



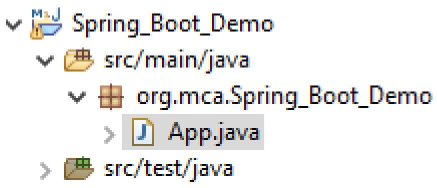
```
<terminated> EmployeeTest (2) [Java Application] C:\Progr
Employee [id=101, name=sonia, salary=0]
Employee [id=101, name=sonia, salary=0]
Employee [id=102, name=sonu, salary=0]
```

Program based on Assignment based Spring Boot and RESTful Web Services

Practical 10

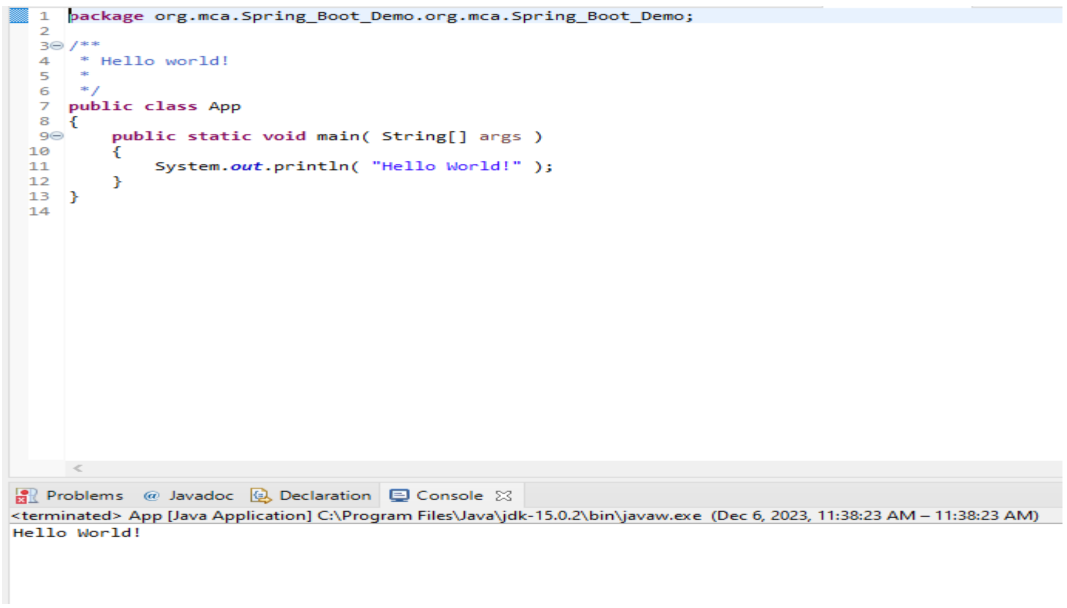
Program no. 1
Aim: Write a program to create a simple Spring Boot application that printsa message.
Line Of Code:

File-> new-> maven project-> default(internal -> Quick start-> 1.1) next->group_ID-> artifact_ID->go for project-> expand the project-> src-> main.JAVA-> org.mca.Spring_Boot_Demo->App.java(open)



```
package org.mca.Spring_Boot_Demo;
/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

Output Screen:



```
package org.mca.Spring_Boot_Demo.org.mca.Spring_Boot_Demo;

/**
 * Hello world!
 *
 */
public class App
{
    public static void main( String[] args )
    {
        System.out.println( "Hello World!" );
    }
}
```

```
<terminated> App [Java Application] C:\Program Files\Java\jdk-15.0.2\bin\javaw.exe (Dec 6, 2023, 11:38:23 AM – 11:38:23 AM)
Hello World!
```

Program based on Assignment based Spring Boot and RESTful Web Services

Program no. 2
Aim: Write a program to demonstrate RESTful Web Services with spring boot.

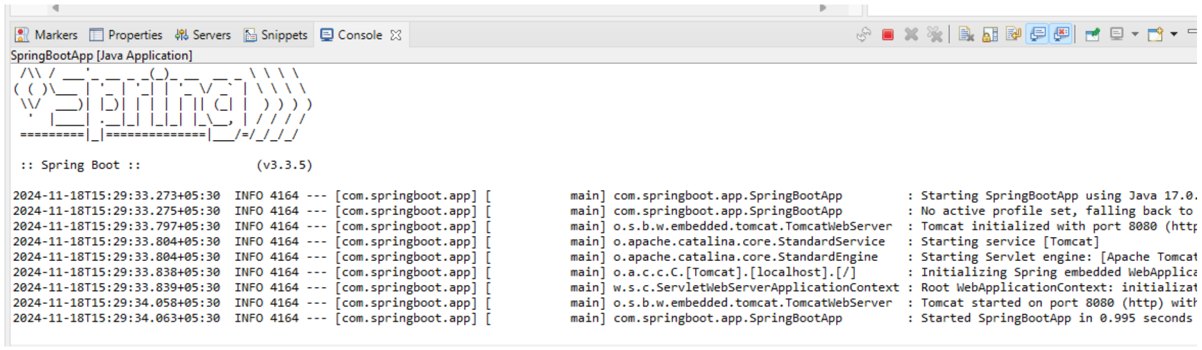
WelcomeController.java

```
package com.springboot.app;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
public class WelcomeController {
@GetMapping("/welcome")
public String welcome()
{
   return "welcome to springboot dear";
}
}
```

SpringBootApp.java

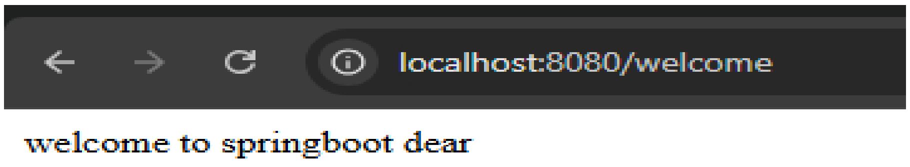
```
package com.springboot.app;

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

@SpringBootApplication
public class SpringBootApp {

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

Output



```
SpringBootApp [Java Application]
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.5)

2024-11-18T15:29:33.273+05:30  INFO 4164 --- [com.springboot.app] [           main] com.springboot.app.SpringBootApp         : Starting SpringBootApp using Java 17.0.
2024-11-18T15:29:33.275+05:30  INFO 4164 --- [com.springboot.app] [           main] com.springboot.app.SpringBootApp         : No active profile set, falling back to
2024-11-18T15:29:33.797+05:30  INFO 4164 --- [com.springboot.app] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http
2024-11-18T15:29:33.804+05:30  INFO 4164 --- [com.springboot.app] [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-11-18T15:29:33.804+05:30  INFO 4164 --- [com.springboot.app] [           main] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat
2024-11-18T15:29:33.838+05:30  INFO 4164 --- [com.springboot.app] [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplica
2024-11-18T15:29:33.839+05:30  INFO 4164 --- [com.springboot.app] [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initializat
2024-11-18T15:29:34.058+05:30  INFO 4164 --- [com.springboot.app] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with
2024-11-18T15:29:34.063+05:30  INFO 4164 --- [com.springboot.app] [           main] com.springboot.app.SpringBootApp         : Started SpringBootApp in 0.995 seconds
```

Navigate to http://localhost:8080/welcome



localhost:8080/welcome

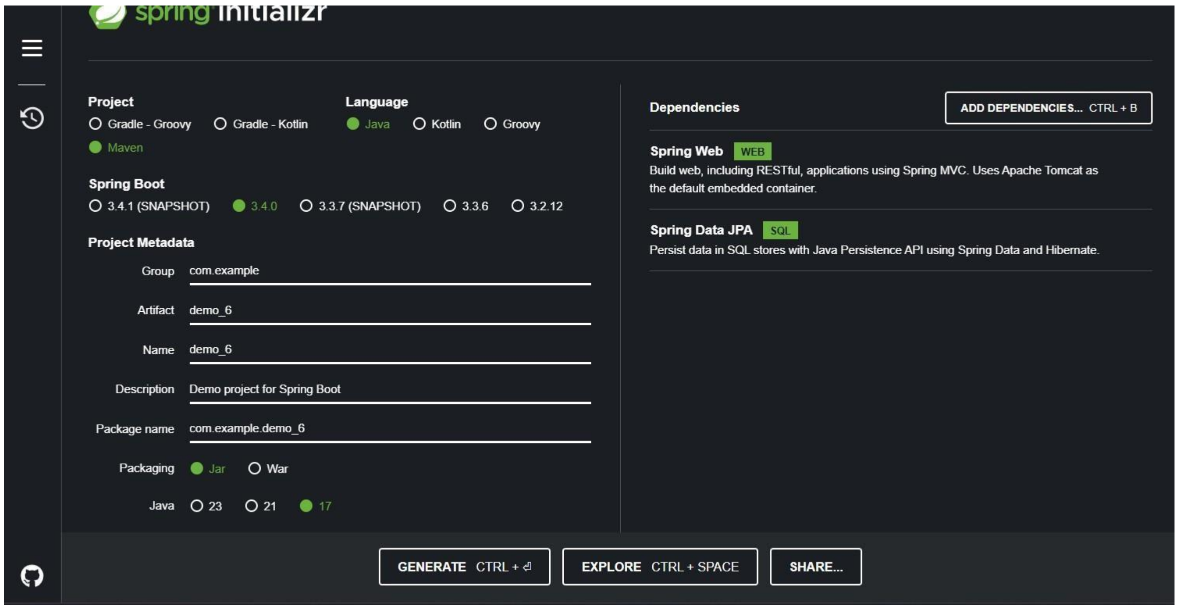
welcome to springboot dear

Roll no.:13 **Jyoti Gupta**

Program based on Assignment based Spring Boot and RESTful Web Services

Program no. 3

Aim: Write a program to demonstrate Database Connection with spring boot.start.spring.io



spring initializr

Project: Gradle - Groovy, Gradle - Kotlin, [✓ Maven]

Language: [✓ Java], Kotlin, Groovy

Spring Boot: 3.4.1 (SNAPSHOT), [✓ 3.4.0], 3.3.7 (SNAPSHOT), 3.3.6, 3.2.12

Project Metadata

Group: com.example

Artifact: demo_6

Name: demo_6

Description: Demo project for Spring Boot

Package name: com.example.demo_6

Packaging: [✓ Jar], War

Java: 23, 21, [✓ 17]

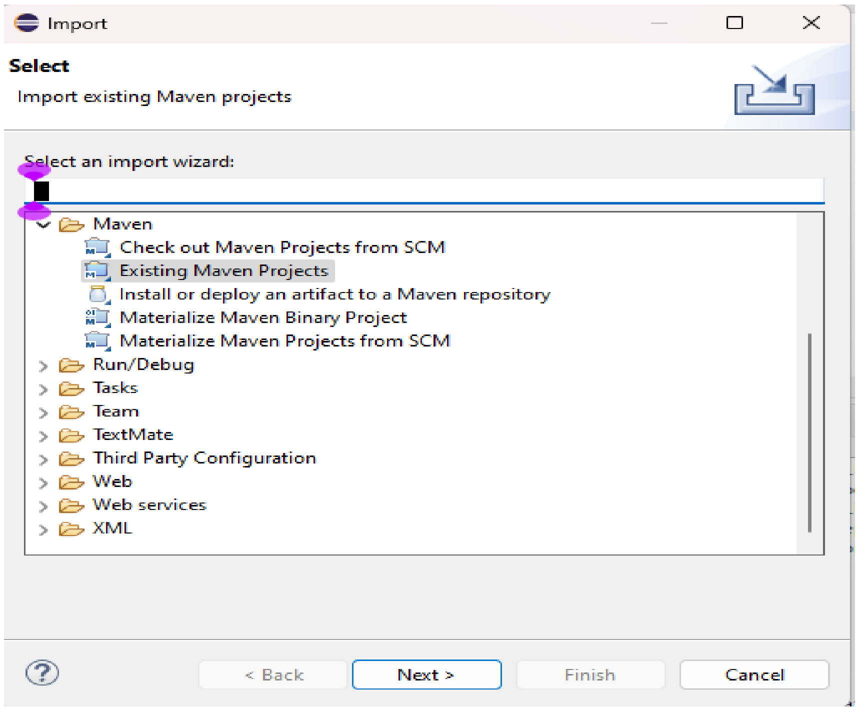
Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

On eclipse

File -> import-> Existing Maven Project



Import

Select

Import existing Maven projects

Select an import wizard:

- Maven
  - Check out Maven Projects from SCM
  - Existing Maven Projects
  - Install or deploy an artifact to a Maven repository
  - Materialize Maven Binary Project
  - Materialize Maven Projects from SCM
- Run/Debug
- Tasks
- Team
- TextMate
- Third Party Configuration
- Web
- Web services
- XML

Roll no.:13 **Jyoti Gupta**

Program based on Assignment based Spring Boot and RESTful Web Services

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>3.4.0</version>
		<relativePath/> <!-- lookup parent from repository -->
	</parent>
	<groupId>com.example</groupId>
	<artifactId>demo_6</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>demo_6</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
		<java.version>17</java.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-data-jpa</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-web</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<scope>test</scope>
		</dependency>
		<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <version>42.6.0</version> <!-- Use the latest version -->
</dependency>

	</dependencies>

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

</project>
```

In src/main/resources

Application.properties

```
spring.application.name=demo_6
# PostgreSQL Database Configuration
spring.datasource.url=jdbc:postgresql://localhost:5433/postgres
spring.datasource.username=postgres
spring.datasource.password=abc
server.port=8081
spring.datasource.driver-class-name=org.postgresql.Driver

# JPA Configuration
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Person.java

```
package    com.example.demo_6;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import                    jakarta.persistence.Id;
```

```
@Entity
public class Person {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    // Getters and Setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

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

Interface PersonRepository

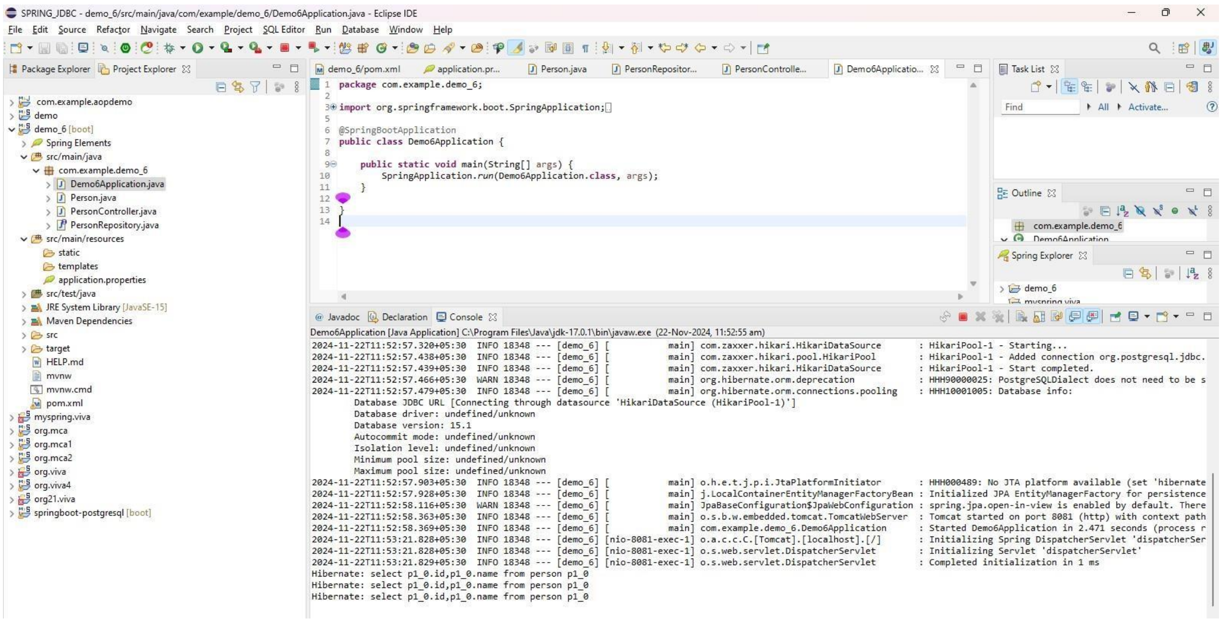
```
package com.example.demo_6;
import org.springframework.data.jpa.repository.JpaRepository;
public interface PersonRepository extends JpaRepository<Person, Long> {
}
```

PersonController.java

```
package com.example.demo_6;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class PersonController {
    @Autowired
    private PersonRepository personRepository;

    @GetMapping("/persons")
    public List<Person> getAllPersons() {
        return personRepository.findAll();
    }
}
```

By default created

```
package com.example.demo_6;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Demo6Application {

        public static void main(String[] args) {

                SpringApplication.run(Demo6Application.class, args);

        } }
```

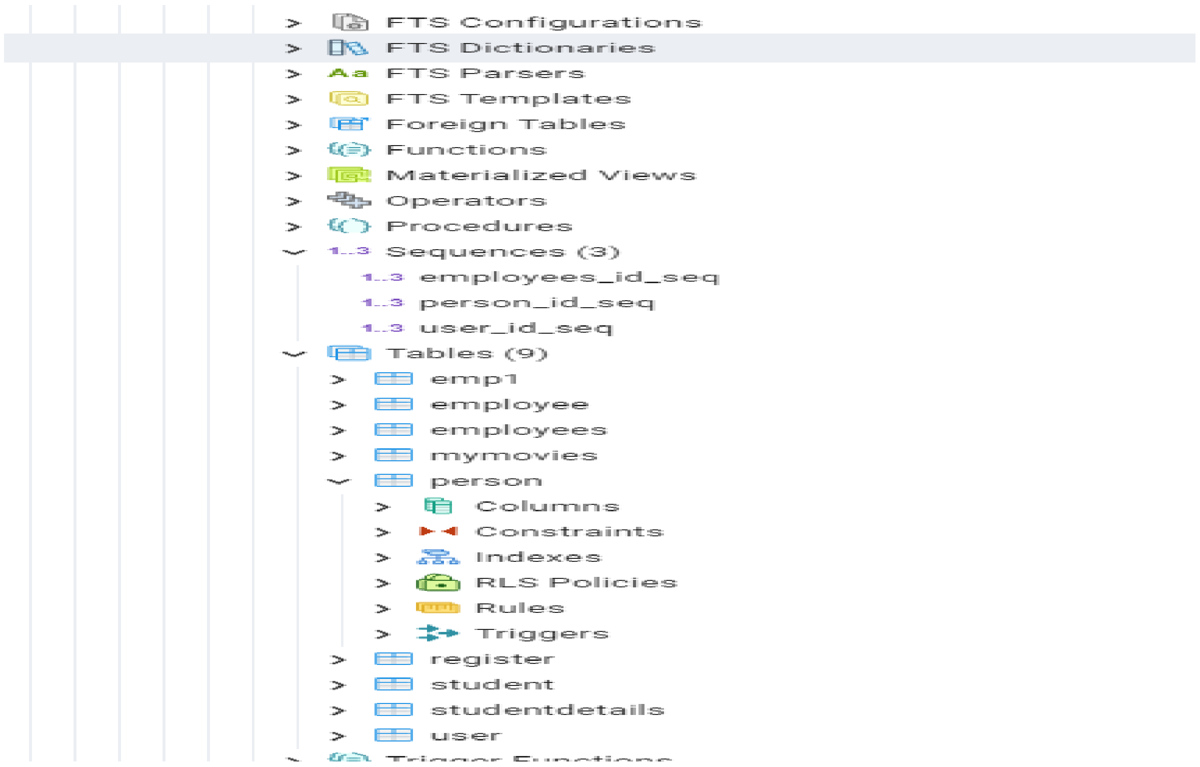


```
package com.example.demo_6;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class Demo6Application {

    public static void main(String[] args) {
        SpringApplication.run(Demo6Application.class, args);
    }

}
```

```
Demo6Application [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (22-Nov-2024, 11:52:55 am)
2024-11-22T11:52:57.320+05:30  INFO 18348 --- [demo_6] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-11-22T11:52:57.438+05:30  INFO 18348 --- [demo_6] [           main] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection org.postgresql.jdbc.
2024-11-22T11:52:57.439+05:30  INFO 18348 --- [demo_6] [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2024-11-22T11:52:57.466+05:30  WARN 18348 --- [demo_6] [           main] org.hibernate.orm.deprecation            : HHH90000025: PostgreSQLDialect does not need to be s
2024-11-22T11:52:57.479+05:30  INFO 18348 --- [demo_6] [           main] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
	Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
	Database driver: undefined/unknown
	Database version: 15.1
	Autocommit mode: undefined/unknown
	Isolation level: undefined/unknown
	Minimum pool size: undefined/unknown
	Maximum pool size: undefined/unknown
2024-11-22T11:52:57.903+05:30  INFO 18348 --- [demo_6] [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate
2024-11-22T11:52:57.928+05:30  INFO 18348 --- [demo_6] [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence
2024-11-22T11:52:58.116+05:30  WARN 18348 --- [demo_6] [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. There
2024-11-22T11:52:58.363+05:30  INFO 18348 --- [demo_6] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8081 (http) with context path
2024-11-22T11:52:58.369+05:30  INFO 18348 --- [demo_6] [           main] com.example.demo_6.Demo6Application      : Started Demo6Application in 2.471 seconds (process r
2024-11-22T11:53:21.828+05:30  INFO 18348 --- [demo_6] [nio-8081-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherSer
2024-11-22T11:53:21.828+05:30  INFO 18348 --- [demo_6] [nio-8081-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2024-11-22T11:53:21.829+05:30  INFO 18348 --- [demo_6] [nio-8081-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
Hibernate: select p1_0.id,p1_0.name from person p1_0
Hibernate: select p1_0.id,p1_0.name from person p1_0
Hibernate: select p1_0.id,p1_0.name from person p1_0
```

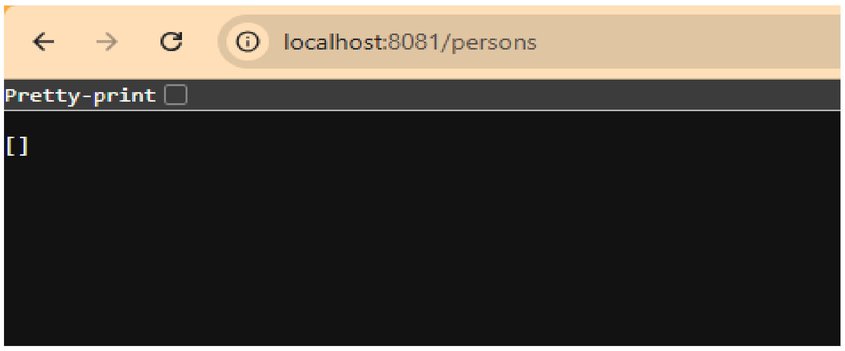
Run DemoApplication as JAVA Application.

Go to PgADMIN - > Run as Administrator -> postgresql 15-> databases ->postgres-> Schemas-> tables -> check Person table is available**.**


```
> FTS Configurations
> FTS Dictionaries
> FTS Parsers
> FTS Templates
> Foreign Tables
> Functions
> Materialized Views
> Operators
> Procedures
v Sequences (3)
    employees_id_seq
    person_id_seq
    user_id_seq
v Tables (9)
    > emp1
    > employee
    > employees
    > mymovies
    v person
        > Columns
        > Constraints
        > Indexes
        > RLS Policies
        > Rules
        > Triggers
    > register
    > student
    > studentdetails
    > user
```

On browser run

http://localhost:8081/persons



```
[]
```

API is working Properly.

Go back to PgADMIN and execute following Queries

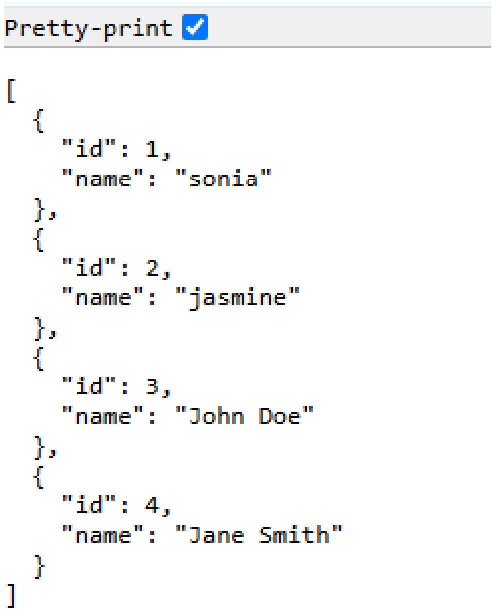
INSERT INTO person (name) VALUES ('sonia');

INSERT INTO person (name) VALUES ('jasmine');

INSERT INTO person (name) VALUES ('John Doe');

INSERT INTO person (name) VALUES ('Jane Smith');

Now again go to browser and run following



```
[
  {
    "id": 1,
    "name": "sonia"
  },
  {
    "id": 2,
    "name": "jasmine"
  },
  {
    "id": 3,
    "name": "John Doe"
  },
  {
    "id": 4,
    "name": "Jane Smith"
  }
]
```