FY MCA Div B

Roll No: 125

Practical Assignment No 1

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

GenericClass:

```
//A generic class to store any type of object

package com.collection.generics;

public class GenericClassExample<T> {
    T obj;
    public GenericClassExample(T obj) {
        // TODO Auto-generated constructor stub
        this.obj = obj;
    }
    //declare an instance method that will return T type object

    T getObject() {
        return obj;
    }
}
```

Main Class:

```
package com.collection.generics;

public class GenericClassTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //Call getObject() method to get Integer object
        Integer i = 20;
        GenericClassExample<Integer> obj = new GenericClassExample<Integer>(i);
        System.out.println("Stored value:" + obj.getObject());

        //In the same way, we will use GenericClassExamplefor storing Double object and retrieve it.
        Double d = 20.25; //same as Double d = new Double(20.25);
        GenericClassExample<Double> obj1 = new GenericClassExample<Double>(d);
        System.out.println("Stored value: " + obj1.getObject());
    }

}
```

Output:

```
Stored value:20
Stored value: 20.25
```

b. Write a Java Program to demonstrate Generic Methods.

GenericMethodClass:

```
package com.collection.generics;

public class GenericMethodDemo {
    //Declaring a generic method
    public <T> void genericMethod(T data) {
        System.out.println("Generic Method.");
        System.out.println("Entered data: " + data);
    }
}
```

Main Class:

```
package com.collection.generics;

public class GenericMethodMain {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        GenericMethodDemo obj = new GenericMethodDemo();
        obj.<String> genericMethod("Sam");
        obj.<Integer> genericMethod(10);
    }

}
```

Output:

```
Generic Method.
Entered data: Sam
Generic Method.
Entered data: 10
```

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

```java
package com.generics.wildcards;

import java.util.Arrays;
import java.util.List;

public class ExampleGenericWildcard {

    private static Number Sum(List<? extends Number> num)
    {
        Double sum = 0.0;
        for(Number n: num)
            sum += n.doubleValue();

        return sum;
    }


    public static void main(String[] args) {
        // TODO Auto-generated method stub
        //Number subtype: INTEGER
        List<Integer> Integer_List = Arrays.asList(1,2,3,4,5);
        System.out.println("List Contains: ");
        System.out.println(Integer_List);
        System.out.println("Sum of the List Elements are: " + Sum(Integer_List));

        //Number subtype: Double
        List<Double> Double_List = Arrays.asList(1.3,2.2,3.5,4.6,5.1);
        System.out.println("List Contains: ");
        System.out.println(Double_List);
        System.out.println("Sum of the List Elements are: " + Sum(Double_List));
    }

}
```

Output:

```
List Contains:
[1, 2, 3, 4, 5]
Sum of the List Elements are: 15.0
List Contains:
[1.3, 2.2, 3.5, 4.6, 5.1]
Sum of the List Elements are: 16.7
```

Practical Assignment No 2

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

```
Package TestJavaCollection;

Import java.util.*;

Public class ArrayLAssignment{

      Public static void main(String[]args){
            //TODOAuto-generatedmethodstub
            ArrayList<String>al=new ArrayList<String>();

            al.add("Sam");
            al.add("Ajay");
            al.add("Niraj");
            al.add("Abhiram");

            System.out.println("Printingtheelementsinlistusingfor--eachloop:");
            for(String var: al) {
                  System.out.println(var);
            }
      }

}
```

Output:

```
Printing the elements in list using for--each loop:
Sam
Ajay
Niraj
Abhiram
```

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

```
Package com.listiterator.example;

Import java.util.*;

Public class BackwardListIterator{

        Public static void main(String[]args){
                //TODOAuto-generatedmethodstub
            ArrayList<String>al=new ArrayList<String>();

            al.add("Sam");
            al.add("Ajay");
            al.add("Niraj");
            al.add("Abhijeet");

            System.out.println("IteratingListinforwarddirection:");
             ListIterator <String>liforward = al.listIterator();

            while(liforward.hasNext()) {
                  System.out.println(liforward.next());
            }

            System.out.println("\n");
            System.out.println("IteratingListinbackwarddirection:");
            ListIterator <String>liback = al.listIterator(al.size());

            while(liback.hasPrevious()) {
                  System.out.println(liback.previous());
            }
      }

}
```

Output:

```
Iterating List in forward direction:
Sam
Ajay
Niraj
Abhijeet


Iterating List in backward direction:
Abhijeet
Niraj
Ajay
Sam
```

Practical Assignment No 3

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

```
Package com.collection.set.example;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.ListIterator;

public class SetExample{

	public static void main(String[]args){
		// TODO Auto-generated method stub
		HashSet<String>obj=new HashSet<String>();

		//additemsinset
		obj.add("Sam");
		obj.add("Bam");
		obj.add("Thank");
		obj.add("you");
		obj.add("Mam");

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

		System.out.println("");
		//convertthesetto List
		List<String>SetToList = new ArrayList<>(obj);
		System.out.println("PrintingHashSetcontentsinreverseorderusingList
Iterator:");
		//printlistinreverseorder
		ListIterator<String>li=SetToList.listIterator(SetToList.size());

		while(li.hasPrevious())
			System.out.println(li.previous());
	}

}
```

Output:

```
Printing HashSet contents using iterator interface:
Mam
Thank
Sam
you
Bam

Printing HashSet contents in reverse order using List Iterator:
Bam
you
Sam
Thank
Mam
```

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

a. Add items in the set.
b. Insert items of one set in to other set.
c. Remove items from the set
d. Search the specified item in the set

```
Package com.collection.set.example;

Import java.util.HashSet;
Import java.util.Iterator;

Public class SetExampleSecond{

      Public static void main(String[]args){
            // TODO Auto-generated method stub
            HashSet<String>obj=new HashSet<String>();

            //additemsinset
            obj.add("Sam");
            obj.add("Bam");

            System.out.println("Set Contains: ");
            Iterator<String>itr=obj.iterator();
            while(itr.hasNext()) {
                  System.out.println(itr.next());
            }

            //insertitemsofonesetintoanotherset
            System.out.println("");
            System.out.println("Copying contents of oneset into  another:");
            HashSet<String>obj1 = new HashSet<String>();
            obj1.add("Thank");
            obj1.add("you");
            obj1.add("Mam");
            obj1.addAll(obj);
            System.out.println("New Set Contains:");
            Iterator<String>it = obj1.iterator();
            while(it.hasNext()) {
                  System.out.println(it.next());
            }

            //removeanitemfromset
            System.out.println("");
            obj1.remove("Mam");
            System.out.println("Set Content safter removing an element:");
            Iterator<String>it1 = obj1.iterator();
            while(it1.hasNext()) {
                  System.out.println(it1.next());
            }

            //Searchthespecifieditеminset
            System.out.println("");
            System.out.println("Searching if set c ontains'Sam'init:");
```

```
            Boolean value=obj.contains("Sam");


             if(value==true)
                    System.out.println("Output:SetContainsSaminit.");
            else
                  System.out.println("Output:SetdoesnotcontainSaminit.");
      }

}
```

Output:

```
Set Contains:
Sam
Bam

Copying contents of one set into another:
New Set Contains:
Mam
Thank
you
Sam
Bam

Set Contents after removing an element:
Thank
you
Sam
Bam

Searching if set contains 'Sam' in it:
Output: Set Contains Sam in it.
```

Practical Assignment No 4

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

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

```
Package com.collection.map.example;

Import java.util.HashMap;
Import java.util.Map;

Public class MapExample{

        Public static void main(String[]args){
                //TODOAuto-generatedmethodstub
            Map<Integer,String>m=new HashMap<Integer,String>();
            //Additemsinmap
            m.put(1, "Sam");
            m.put(2,"Bam");
            System.out.println("Map contains:"+m);

            //getvalueofspecifiedkey
            System.out.println("");
            System.out.println("Getting value of specified key:");
            String value = m.get(1);
            System.out.println("Value stored at key1:"+value);

            //InsertElementsofMapintoanotherMap.
            System.out.println("");
            System.out.println("Inserting Elements of onemap into another  map:");
            Map <Integer, String>m1 = new HashMap<Integer, String>();
            m1.put(1,"Thank");
            m1.put(2,"you");
            m1.put(3,"Mam");
            m1.putAll(m);
            System.out.println("All key and values of Map1 are:"+m1);

            //remove items from map
            System.out.println("");
            System.out.println("Removing item form Map:");
            m1.remove(2);
            System.out.println("After removing element Map1 contains:"+m1);

            //searchspecifickeyinmap
            System.out.println("");
            System.out.println("Searching if Map contains 'Sam' init:");
            booleanvalue1=m1.containsValue("Sam");
```

```
                if(value1==true)



            else  System.out.println("Output: Map Contains Sam in it.");

                  System.out.println("Output:Map does not contain Sam  init.");

      }

}
```

Output:

```
Map contains: {1=Sam, 2=Bam}

Getting value of specified key:
Value stored at key 1: Sam

Inserting Elements of one map into another map:
All key and values of Map1 are: {1=Sam, 2=Bam, 3=Mam}

Removing item form Map:
After removing element Map1 contains: {1=Sam, 3=Mam}

Searching if Map contains 'Sam' in it:
Output: Map Contains Sam in it.
```

Practical Assignment No 5

a. Write a Java program using Lambda Expression to print “Hello, India!”.

```
Package com.example.assignment;

@FunctionalInterface
Public interface Q2LambdaHelloIndia{
      //declaringanabstractfunction
      Public void printHelloIndia();
}


Package com.example.assignment;

Public class Q2Lambda{

      Public static void main(String[]args){
            //TODOAuto-generatedmethodstub
            Q2LambdaHelloIndiaobj=()->System.out.println("Hello,India!");
            obj.printHelloIndia();
      }

}
```

Output:

```
Hello, India!
```

b. Write a Java program using Lambda Expression with single parameters.

```
Package com.lambda.example;

@FunctionalInterface
Interface lambdaExampleInterface{
      Void print(Stringmessage);
}

Public class LambdaSingleParameter{
      Public static void main(String[]args){
            lambdaExampleInterfaceprintMessage=(message)->System.out.println(message);
            printMessage.print("Hello India!");

      }

}
```

Output:

```
Hello India!
```

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

File1:

```
Package com.example.assignment;

@FunctionalInterface
Public interface Q4AddLambda{
      Public void add(int num1,int num2);
}
```

File2:

```
Package com.example.assignment;

@FunctionalInterface
Public interface Q4MultiplyLambda{
      Public void multiply(int num1,int num2);
}
```

Main File:

```
Package

com.example.assignment;

 import java.util.Scanner;

public class Q4Lambda {

      public static void main(String[]args){
            try(//TODOAuto-generatedmethodstub Scanner s
            = new Scanner(System.in)) {
                  System.out.println("Enter First Number: ");
                  int num1 = s.nextInt();
                  System.out.println("EnterSecondNumber:"); int
                  num2 = s.nextInt();


                  Q4AddLambdaobj1=(value1,value2)->{
                        System.out.println("Additionofgivennumbersis:"+(value1+
value2));
                  };
                  Q4MultiplyLambda obj2 = (value1, value2)->{
                        System.out.println("Multiplicationofgivennumbersis:"+value1*
value2);
                  };

                  obj1.add(num1, num2);
                  obj2.multiply(num1,num2);
            }
      }

}
```

Output:

```
Enter First Number:
5
Enter First Number:
3
Addition of given numbers is: 8
Multiplication of given numbers is: 15
```

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

a. Convert Fahrenheit to Celsius

b. Convert Kilometres to Miles.

```
Package com.lambda.example;

@FunctionalInterface
Interface Convert{
      Double convertInput(double input);
}

Public class LambdaConvert{

      Public static void main(String[]args){
            ConverttoCelsius=(fahrenheit)->(fahrenheit-32)*5.0/9.0; Convert
            toMiles = (kilometers) ->kilometers * 0.621371;


            //Executethelambdaexpressions
            Double celsius=toCelsius.convertInput(100);
            System.out.println("100degrees Fahrenheit is "+ celsius + " degrees Celsius.");
            Double miles=toMiles.convertInput(100);System.out.println("100kilometersis "+
miles + " miles.");
      }
}
```

Output:

```
100 degrees Fahrenheit is 37.77777777777778 degrees Celsius.
100 kilometers is 62.137100000000004 miles.
```

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

```
Package com.lambda.example;

@FunctionalInterface
Interface Add{
      Int add(int a,int b);
}

@FunctionalInterface
Interface Subtract{
      Int subtract(int a,int b);
}
Public class LambdaWithoutReturn{
      Public static void main(String[]args){
            Add adding = (a, b) ->a + b;
            Subtractsubtracting=(a,b)->a-b;

            int sum = adding.add(155, 9);
            System.out.println("Thesumis:"+sum);
            int difference = subtracting.subtract(124, 2);
            System.out.println("Thedifferenceis:"+difference);
      }
}
```

Output:

```
The sum is: 164
The difference is: 122
```

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

```
Package com.lambda.example;

@FunctionalInterface
Interface ConcatenateInterface{
      Stringconcatenate(Stringa,Stringb);
}
Publicclass LambdaConcatenate {

      Public static void main(String[]args){
            ConcatenateInterfaceconcat=(Stringa,Stringb)->a+b;

            Stringstr1 = "Hello,";
            Stringstr2="India!";
            Stringresult=concat.concatenate(str1,str2);

            System.out.println(result);
      }

}
```

Output:

```
Hello, India!
```

Practical Assignment No 6

a. Write Programs to demonstrate different Implicit Objects

- **Out**
- **Request**
- **Session**

```
<%@ page language="java" contentType="text/html;charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPEhtml>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>OutObject</h1>
<% out.println("Luffy:Thisis...aloveordeal");%>
<h1>ReuqestObject</h1>
<%
      String uri=request.getRequestURI();
      out.println("RequestedURI:"+uri);
%>
<h1>Session Object</h1>
<%
      session.setAttribute("luffy", "I refuse your refusal");
      Stringattribute=(String)session.getAttribute("luffy");
      out.println("Thevalueofthesessionattribute'attribute'is:"+ attribute);
%>
</body>
</html>
```

Output:

Out Object

Luffy : This is... a love ordeal

Reuqest Object

Requested URI: /MCA-37/CoreTag.jsp

Session Object

The value of the session attribute 'attribute' is: I refuse your refusal

b. Write Programs to demonstrate temporary storage using Bean.

```
<%@ page import="java.util.ArrayList"%>
<jsp:useBean id="myBean" class="jspExample.MyBean" scope="request"/>
<%
    //Setdatainthebean
    myBean.setData("Sorry,butitlookslikeI'mdead.");

    // Retrieve data from the bean
    String data=myBean.getData();
%>
<html>
<head><title>TemporaryStorageUsingBean</title></head>
<body>
    <h2>DatastoredinBean:</h2>
    <p><%=data%></p>
</body>
</html>
```

Output:

Data stored in Bean:

Sorry, but it looks like I'm dead.

c. Write a program to demonstrate Standard Action tags.

```
<%@ page language="java" contentType="text/html;charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPEhtml>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Practical7</title>
</head>
<body>
<body>
<%@include file="header.jsp"%><!--Directivetoincludeheader-->

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

<%--JSPScriptlet--%>
<%
    count++;
    out.println("This is a Example of scriptlet.Countisnow:"+count);
%>


<%--JSPExpression--%>
<p>This is an Example of Directiveexpression.Thevalueofcountisnow: <%= count
%></p>

<%@ include file="footer.jsp"%><!--Directivetoincludefooter-->
</body>
</body>
</html>
```

Output:

d. Write a program to demonstrate JSP Directives.

```
<%@ page language="java" contentType="text/html;charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ include file="header.jsp"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPEhtml>
<html>
<head>
<meta charset="ISO-8859-1">
<title>JSPDirectives</title>
</head>
<body>
    <h2>WelcometoJSPDirectives!</h2>
    <c:out value="${'Istillhavemyfriends!'}"/>
    <%@ include file="footer.jsp"%>
</body>
</html>
```

Output:

e. Write a program to demonstrate Session Tracking using Cookies.

```
<%@ page import="java.io.PrintWriter"%>
<%
    // Get the current session or create a new one
    HttpSessionsession1=request.getSession(true);

    // Set session attribute
    session1.setAttribute("username", "Session:luffy");

    //Createacookiefortheusername
    CookieusernameCookie=new Cookie("username","Cookie:Luffy");
    response.addCookie(usernameCookie);
%>

<html>
<head><title>Session Tracking Using Cookies</title></head>
<body>
    <h2>Session Tracking Using Cookies</h2>
    <p>Username stored in session:<%=session1.getAttribute("username")
%></p>
    <p>Username stored in cookie:<%=usernameCookie.getValue()%></p>
</body>
</html>
```

Output:

Session Tracking Using Cookies

Username stored in session: Session:luffy

Username stored in cookie: Cookie:Luffy

f. Write a program to demonstrate JSTL Tags.

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

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

<html>
<head>
    <title>JSTLDemo</title>
</head>
<body>
    <h2>JSTLCoreTagsDemo</h2>

    <c:set  var="message" value="Iloveheroes,butIdon'twanttobeone."

/>

    <p>Message:<c:out value="${message}"/></p>

    <c:if test="${5>3}">
        <p>Theconditionistrue.</p>
    </c:if>

    <c:forEach var="i" begin="1" end="5">

        <p>Number:${i}</p>
    </c:forEach>

</body>
</html>
```

Output:

JSTL Core Tags Demo

Message: I love heroes, but I don't want to be one.

The condition is true.

Number: 1

Number: 2

Number: 3

Number: 4

Number: 5

JSTL Formatting Tags Demo

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

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<%@ page language="java" contentType="text/html;charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPEhtml>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Practical1</title>
<style>
body{
    font-family:Arial,sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding:0;
}
h1{
    color:#333;
    text-align:center;
    margin-top: 20px;
}

form{
    text-align:center;
    margin-top: 20px;
}
table{
    margin: 0 auto;
    margin-top:20px;
    border-collapse:collapse;
    width: 80%;
}
table,th,td{
    border:1pxsolid#ddd;
    padding: 8px;
}
th{
    padding-top: 12px;
    padding-bottom: 12px;
    text-align: left;
    background-color:purple;
    color: white;
}


input{
height : 20px;
padding:5px10px;
```

```
}

</style>

</head>
<body>
      <h1>Addanewentry</h1>
    <form method="get">
        <label for="search">Search:</label>
        <input type="text" id="search" name="search" placeholder="searchby
name">
    </form>



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

    <sql:query dataSource = "${snapshot}" var =
         "result">
    SELECT*from telephone where name LIke ?;
         <sql:param value="%${param.search}%"/>

      </sql:query>

      <table  border="1" width="100%">
         <tr>
            <th>Id</th>
            <th>Name</th>
            <th>PhoneNUmber</th>
         </tr>

         <c:forEach var="row" items="${result.rows}">
            <tr>
               <td><c:out value="${row.id}"/></td>
               <td><c:out value="${row.name}"/></td>
               <td><c:out value="${row.phoneNumber}"/></td>
            </tr>
         </c:forEach>
      </table>

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

Output:

Add a new entry

Search: search by name

Id	Name	Phone NUmber
1	raj	1112223333
2	abhishek	1112223333
3	Shreya	989898998
4	Abdul	2424242424
5	Bhushan	323232323

Add a new entry

Search: search by name

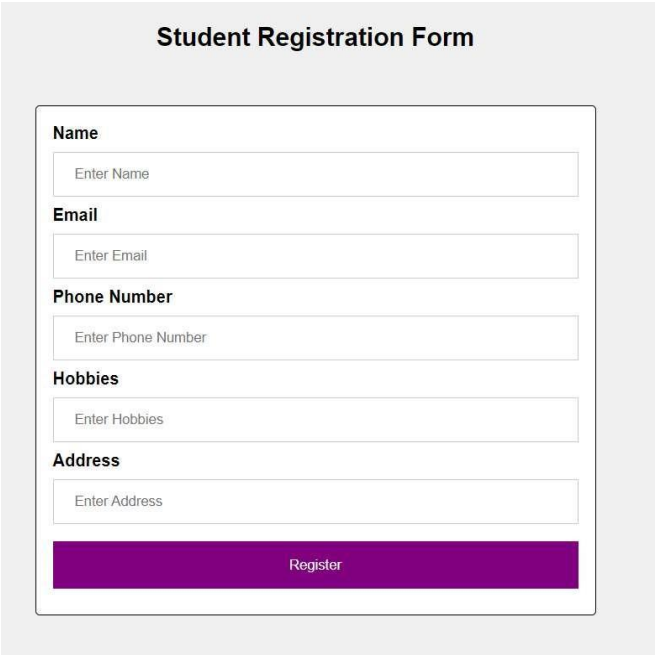
Id	Name	Phone NUmber
3	Shreya	989898998

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

```
<%@ page language="java" contentType="text/html;charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<!DOCTYPEhtml>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Practical2</title>
<style>
body{
        font-family:Arial,sans-
        serif;
         background-color:
        #f0f0f0;
}
.container{
        width: 500px;
        padding:16px;
        background-
        color:white; margin: 0
        auto;
        margin-top:50px; border:
        1pxsolidblack;
        border- radius:4px;
}
input[type=text], input[type=password] { width:
        100%;  padding: 12px 20px;
        margin:8px0; display:
        inline-block;
        border:1pxsolid#ccc
        ; box-sizing:border-
        box;
}
button{
        background-color:purple;color:
        white;
        padding:14px20px;
        margin: 8px 0;
        border: none;
        cursor: pointer;
        width: 100%;
}
button:hover{
        opacity:0.8;
}
```

```
    H2{ text-align:center;
  }

  </style>
  </head>
  <body>
  <h2>StudentRegistrationForm</h2>
  <div class="container">
      <label for="name"><b>Name</b></label>
      <input type="text" placeholder="EnterName" name="name" required>

      <label for="email"><b>Email</b></label>
      <input type="text" placeholder="EnterEmail"  name="email" required>

      <label for="phone"><b>PhoneNumber</b></label>
      <input type="text" placeholder="EnterPhoneNumber"  name="phone" required>

      <label for="hobbies"><b>Hobbies</b></label>

      <input type="text" placeholder="EnterHobbies" name="hobbies" required>

      <label for="address"><b>Address</b></label>

      <input type="text" placeholder="EnterAddress"  name="address" required>

      <button  type="submit">Register</button>

  </div>
  </body>
  </html>
```

Output:



Practical Assignment No 7

a. Write a program to print Singer Name and Age using spring framework.

Singer.java

```
Package com.example.SpringTest;

Public class Singer{
      String name;
      int age;
      Public String getName(){
      returnname;
      }
      Public void setName(String name){
      this.name=name;
      }
      Public int getAge(){
      Return age;
      }
      Public void setAge(int age){
      this.age=age;
      }
      Void displayInfo()
      {
            System.out.println("Name:"+name+"Age:"+age);
      }
}
```

ApplicationContext.xml

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

      <bean id="Singer" class="com.example.Spring">
            <property name="name" value="Luffy"></property>
            <property name="age" value="19"></property>
      </bean>
</beans>
```

SingerTest.java

```
Package com.example.SpringTest;
Import org.springframework.context.ApplicationContext;
Import org.springframework.context.support.ClassPathXmlApplicationContext;

Public class SingerTest{
     Private static ApplicationContextctx;
      public static void main(String[] args) {
            ApplicationContext context = new
            ClassPathXmlApplicationContext("Appctx.xml");
            Singertemp=(Singer)ctx.getBean("Singer");
            s1.displayInfo();
      }
}
```

Output:

```
Console ×
<terminated> test [Java Application] C:\Users\Raj\.p2\poc
Singer [name=Luffy, age=19]
```

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

PojoClass

```
Package MCA;

Public class Zoro{
        private String name;
        private double height;
        private int swords;


//      setterandgettermethods
            Public StringgetName(){
                Return name;
        }
       Public void setName(String name){
                this.name=name;
        }
        Public double getHeight(){
                Return height;
        }
        Public void setHeight(double height){
                this.height=height;
        }
        Public int getSwords(){
                Return swords;
        }
        Public void setSwords(int swords){
                this.swords=swords;
        }


//      Constructor
        Public Zoro(String name,double height,int swords){
                super();
                this.name
                = name;
                this.height=height;
                this.swords=swords;
        }
        Public Zoro(){
                super();
        }

//      tostring
        method
        @Override
        Public StringtoString(){
                return"nameofCharacter="+name+",heightofCharacter="
+height+",No.ofswords="+swords;
        }
}
```

ApplicationContext.xml

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


<bean class="MCA.Zoro" name="zoro" p:name="PirateHunterRoronoaZoro"
p:height="6.2" p:swords="3"/>
  </beans>
```

MainClass

```
Package MCA;
Import org.springframework.context.ApplicationContext;
Import org.springframework.context.support.ClassPathXmlApplicationContext;

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

                ApplicationContextcontext=new
ClassPathXmlApplicationContext("MCA/mcaConfig.xml"
                );
                Zoro temp = (Zoro)
                context.getBean("zoro");
                System.out.println(temp);
        }
}
```

MavenDependencies

Output:

```
Console ×
<terminated> test [Java Application] C:\Users\Raj\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin
name of Character = Pirate Hunter Roronoa Zoro, height of Character = 6.2, No. of swords = 3
```

c. Write a program to demonstrate dependency injection via Constructor. (Primitive)

PojoClass

```
Package MCA;

Public class luffy{
        private String name;
        private int gears;
        private double
        height;
        public luffy(String name,int gears,double height){
                super();
                this.name
                = name;
                this.gears=gears;
                this.height=height;
        }
        @Override
        Public StringtoString(){
                return"Charactername="+name+",No.ofgears="+ gears + ",
height = "+ height + "]";
        }
}
```

ApplicationContext.xml

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

xsi:schemaLocation="http://www.springframework.org/schema/be
anshttp://www.springframework.org/schema/beans/spring-
beans.xsdhttp://www.springframework.org/schema/contexthttp://w
ww.springframework.org/schema/context/spring-context.xsd">



<bean class="MCA.luffy" name="luffy"  c:name="MonkeyD.Luffy" c:height="5.8" c:gears="5"/>

 </beans>
```

MainClass

```
Package MCA;
Import org.springframework.context.ApplicationContext;
Import org.springframework.context.support.ClassPathXmlApplicationContext;

Public class test{
        Public static void main(String[]args){
             ApplicationContextcontext=new
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");
             luffytemp=(luffy)context.getBean("luffy");
             System.out.println(temp);
        }
}
```

Output:

```
Console ×
<terminated> test [Java Application] C:\Users\Raj\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_1
 Charactername = Monkey D. Luffy, No. of gears = 5, height = 5.8]
```

d. Write a program to demonstrate dependency injection via setter method. (Non-Primitive)

PojoClass

```
Package MCA;

Public class sanji{

        private String name;
        private double height;
        private Zoro obj;


        public StringgetName(){
                return name;
        }
       Public void setName(String name){
                this.name=name;
        }
        Public double getHeight(){
                Return height;
        }
        Public void setHeight(double height){
                this.height=height;
        }
        Public ZorogetObj(){
                Return obj;
        }
        Public void setObj(Zoro obj){
                this.obj=obj;
        }

        Public sanji(String name,double height,Zoro obj){
                super();
                this.name
                = name;
                this.height=height;
                this.obj=obj;
        }

        Public  sanji(){
                super();
                //TODOAuto-generatedconstructorstub
        }
        @Override
        Public StringtoString(){
                return"sanji[name="+name+",height="+height+",\nobj="
  +obj+"]";
        }

}
```

ReferenceClass

```
Package MCA;

Public class Zoro{
        private String name;
        private doubleheight;
        private int swords;


  //     setterandgettermethods
                Public String getName(){
                    Return name;
        }
        Public void setName(Stringname){
                this.name=name;
        }
        Public double getHeight(){
                Return height;
        }
        Public void setHeight(double height){
                this.height=height;
        }
        Public int getSwords(){
                Return swords;
        }
        Public void setSwords(int swords){
                this.swords=swords;
        }


    //      Constructor
        Public Zoro(String name,double height,int swords){
                super();
                this.name
                = name;
                this.height=height;
                this.swords=swords;
        }
        Public Zoro(){
                super();
        }

    //      tostring

        method

        @Override
        Public StringtoString(){
                Return "nameofCharacter="+name+",heightofCharacter="
    +height+",No.ofswords="+swords;
        }
}
```

ApplicationContext.xml

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


<bean class="MCA.Zoro" name="zoro" p:name="PirateHunterRoronoaZoro"
p:height="6.2"  p:swords="3"/>
<bean class="MCA.sanji" name="sanji" p:name="VinsmokeSanji" p:height="6.0"  p:obj-ref="zoro"/>

 </beans>
```

MainClass

```
Package MCA;

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

Public class test{

        Public static void main(String[]args){


            ApplicationContextcontext=new
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");
            sanjitemp=(sanji)context.getBean("sanji");
            System.out.println(temp);
      }

}
```

Output:

```
Console ×
<terminated> test [Java Application] C:\Users\Raj\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe
sanji [name=Vinsmoke Sanji, height=6.0,
obj=name of Character = Pirate Hunter Roronoa Zoro, height of Character = 6.2, No. of swords = 3]
```

e. Write a program to demonstrate dependency injection via Constructor. (Non-Primitive) By Ref

PojoClass

```
Package MCA;

Public class ussop{

        private String Name;
        private double height;
        private luffy obj;

        @Override
        publicStringtoString(){
                return "ussop[Name="+Name+",height="+height+",\nobj="
  +obj+"]";
        }

        Public ussop(String name,double height,luffy obj){
                super(
                );Nam
                e=
                name;
                this.height=height;
                this.obj=obj;
        }

}
```

ReferenceClass

```
Package MCA;


Public class luffy{

        private String name;
        private int gears;
        private double
        height;


        publicluffy(Stringname,intgears,doubleheight){

                super();
                this.name
                = name;

                this.gears=gears;

                this.height=height;
        }

        @Override
        Public StringtoString(){

                return"Charactername="+name+",No.ofgears="+ gears + ",
height = "+ height + "]";
        }

}
```

ApplicationContext.xml

```
<?xml version="1.0"encoding="UTF-8"?>

<beans
xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
xmlns:c="http://www.springframework.org/schema/c"
xsi:schemaLocation="http://www.springframework.org/schema/be
anshttp://www.springframework.org/schema/beans/spring-
beans.xsdhttp://www.springframework.org/schema/contexthttp://w
ww.springframework.org/schema/context/spring-context.xsd">


<bean class="MCA.luffy" name="luffy" c:name="MonkeyD.Luffy" c:height="5.8" c:gears="5"/>

<bean  class="MCA.ussop" name="ussop" c:name="SogekingUssop" c:height="5.11" c:obj-ref="luffy"/>


 </beans>
```

MainClass

```
Package  MCA;
Import org.springframework.context.ApplicationContext;
Import org.springframework.context.support.ClassPathXmlApplicationContext;

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

                ApplicationContextcontext=new
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");
                ussoptemp=(ussop)context.getBean("ussop");
                System.out.println(temp);
        }
}
```

Output:

```
Console ×
<terminated> test [Java Application] C:\Users\Raj\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230
ussop [Name=Sogeking Ussop, height=5.11,
obj= Charactername = Monkey D. Luffy, No. of gears = 5, height = 5.8]]
```

f. Write a program to demonstrate dependency injection via Constructor. (Collection)

PojoClass

```
Package MCA;
Import ava.util.*;

Public class strawHat{
        Private Stringname;
        private List<String>
        crewName;
        private Set<String> bounty;
        private Map<String, String>
        ability;

        public strawHat(Stringname,List<String>crewName,Set<String>bounty,
Map<String, String>ability) {
                super();
                this.name
                = name;
                this.crewName=crewName;
                this.bounty = bounty;
                this.ability = ability;
        }

        @Override
        Public StringtoString(){
                Return "strawHat[name="+name+",\ncrewName="+crewName+",
\nbounty="+bounty+",\nability="+ability+"]";
        }


}
```

ApplicationContext.xml

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


<bean  class="MCA.strawHat"  name="strawHat">
<constructor-arg name="name" value="TheStrawHatPirates"/>

<constructor-arg name="crewName">
<list>
<value>MonkeyD.Luffy</value>
<value>RoronoaZoro</value>
<value>FirstsonofseaJimbei</value>
<value>VinksmokeSanji</value>
<value>DemonchildNicoRobin</value>
```

```
</list>
</constructor-arg>

<constructor-arg name="bounty">

<set>
<value>3,000,000,000</value>
<value>1,200,000,000</value>
<value>1,100,000,000</value>
<value>1,032,000,000</value>
<value>930,000,000</value>
</set>
</constructor-arg>

<constructor-arg name="ability">

<map>
<entry key="luffy"  value="rubberbody"/>

<entry key="zoro" value="swordsman"/>

<entry key="jimbei" value="Helmsman"/>

<entry key="sanji" value="cook"/>

<entry key="robin" value="archaeologist"/>

</map>
</constructor-arg>

</bean>
</beans>
```

MainClass

```
Package MCA;
Import org.springframework.context.ApplicationContext;
Import org.springframework.context.support.ClassPathXmlApplicationContext;

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

                ApplicationContextcontext=new
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");
                strawHattemp=(strawHat)context.getBean("strawHat");
                System.out.println(temp);


        }
}
```

:

```
Console ×
<terminated> test [Java Application] C:\Users\Raj\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (21 Dec 2023, 18:14:29 – 18:14:29)
strawHat [name=The Straw Hat Pirates,
crewName=[Monkey D. Luffy, Roronoa Zoro, First son of sea Jimbei, Vinksmoke Sanji, Demon child Nico Robin],
bounty=[3,000,000,000, 1,200,000,000, 1,100,000,000, 1,032,000,000, 930,000,000],
ability={luffy=rubber body, zoro=swordsman, jimbei=Helmsman, sanji=cook, robin=archaeologist}]
```

g. Write a program to demonstrate Auto wiring.

PojoClass

```
Package MCA;

Public class chopper{

        private ZoroZoro;

        public ZorogetZoro(){
                return Zoro;
        }

        Public void setZoro(Zorozoro){ Zoro=zoro;
        }

        Public chopper(MCA.Zorozoro){
                super(
                );
                Zoro=
                zoro;
        }

        Public chopper(){
                super();
        }

        @Override
        Public StringtoString(){
                return"chopper[Zoro="+Zoro+"]";
        }
}
```

ReferenceClass

```
Package MCA;

Public class Zoro{
        private String name;
        private double height;
        private int swords;


  //     setterandgettermethods
                Public StringgetName(){
                        Return name;
        }
        Public void setName(String  name){
                        this.name=name;
}
        Public double getHeight(){
                        Return height;
        }
        Public void setHeight(doublevheight){
                        this.height=height;
        }
        Public int getSwords(){
                        Return swords;
        }
```

```
        Public void setSwords(int swords){
                this.swords=swords;
        }


//      Constructor
        Public Zoro(String name,double height,int swords){
                super();
                this.name
                = name;
                this.height=height;
                this.swords=swords;
        }
        Public Zoro(){
                super();
        }

//      tostring

        method

        @Override
        public StringtoString(){
                return  "nameofCharacter="+name+",heightofCharacter="
+height+",No.ofswords="+swords;
        }

}
```

ApplicationContext.xml

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


<bean class="MCA.Zoro" name="Zoro" p:name="PirateHunterRoronoaZoro"
p:height="6.2" p:swords="3"/>
<bean class="MCA.chopper" name="chopper" autowire="byType"/>

 </beans>
```

MainClass

```
Package MCA;
Import org.springframework.context.ApplicationContext;
Import org.springframework.context.support.ClassPathXmlApplicationContext;

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

```
            ApplicationContextcontext=new
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");
            choppertemp=(chopper)context.getBean("chopper");
             System.out.println(temp);
      }
}
```

Output:

```
Console ×
<terminated> test [Java Application] C:\Users\Raj\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (21 Dec 2023, 19:48:35 – 19:48:35) [pid: 6564]
chopper
[Zoro=name of Character = Pirate Hunter Roronoa Zoro, height of Character = 6.2, No. of swords = 3]
```

Practical Assignment No 8

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

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/4.0.0http://maven.apache.org/xsd/maven-
4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

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

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

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

<dependencies>

        <!--https://mvnrepository.com/artifact/org.springframework/spring-core-->
        <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>5.2.3.RELEASE</version>
        </dependency>

          <!--https://mvnrepository.com/artifact/org.springframework/spring-context-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.3.RELEASE</version>
        </dependency>

        <!--https://mvnrepository.com/artifact/org.springframework/spring-aop-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.2.3.RELEASE</version>
        </dependency>

          <!--https://mvnrepository.com/artifact/org.aspectj/aspectjrt-->
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>1.9.7</version>
        </dependency>

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

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

Interface

```
Package aop;

Public interface Guitar{

        Public void makeSong();

}
```

TargetObject

```
Package aop;

Public class brook implements Guitar{


        Public void makeSong(){

                System.out.println("Song
                Started");
                System.out.println("Song Ended");

        }


}
```

AspectClass

```
Package aop;

import org.aspectj.lang.annotation.After;
import
org.aspectj.lang.annotation.AfterReturning;
import
org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;

@Aspect
Public class mcaAspect{

        @Before("execution(*brook.makeSong())")
        Public void beforeSong(){
                System.out.println("YahooYahoo:IambeforeAspect");
        }
}
```

ConfigurationClass

```
<?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
" xmlns:aop="http://www.springframework.org/schema/aop"
```

```
xsi:schemaLocation="http://www.springframework.org/schema/bea
nshttp://www.springframework.org/schema/beans/spring-
beans.xsdhttp://www.springframework.org/schema/aophttp://ww
w.springframework.org/schema/aop/spring-aop.xsd">


<aop:aspectj-autoproxy/>
<bean name="brook" class="aop.brook"/>
<bean name="mcaaspect"  class="aop.mcaAspect"/>

</beans>
```

MainClass

```
Package aop;

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

Public class App{

        public static void main(String[] args) { ApplicationContext
                context= new
ClassPathXmlApplicationContext("aop/aopConfig.xml");
                Guitar temp = (Guitar)
                context.getBean("brook"); temp.makeSong();

        }

}
```

Output:

```
Console ×
<terminated> App [Java Application] C:\Users\Raj\.p2\pool\plug
Yahoo Yahoo : I am before Aspect
Song Started
Song Ended
```

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

AspectClass

```
Package aop;

import org.aspectj.lang.annotation.After;
import
org.aspectj.lang.annotation.AfterReturning;
import
org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;

@Aspect
Public class mcaAspect{

        @After("execution(*brook.makeSong())")
        Public void afterSong(){
                System.out.println("YahooYahoo:IamAfterAspect");
        }
}
```

Output:

```
Console ×
<terminated> App [Java Application] C:\Users\Raj\.p2\poo
Song Started
Song Ended
Yahoo Yahoo : I am After Aspect
```

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

AspectClass

```
Package  aop;

import org.aspectj.lang.annotation.After;
import
org.aspectj.lang.annotation.AfterReturning;impo
rt org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;

@Aspect
Public class mcaAspect{

        @Around("execution(*brook.makeSong())")
        Public void aroundSong(){
                System.out.println("YahooYahoo:AroundAspect");
        }
}
```

Output:

```
Console ×
<terminated> App [Java Application] C:\Users\Raj\.p2\p
Yahoo Yahoo : Around Aspect
```

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

AspectClass

```
Package aop;

import org.aspectj.lang.annotation.After;
import
org.aspectj.lang.annotation.AfterReturning;
import
org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Around;
import
org.aspectj.lang.annotation.Aspect;
import
org.aspectj.lang.annotation.Before;
import
org.aspectj.lang.annotation.Pointcut;

@Aspect
Public class mcaAspect{


        @AfterReturning("execution(*brook.makeSong())")
        Public void AfterReturnSong(){
                System.out.println("YahooYahoo:AfterReturingAspect");
        }


}
```

Output :

```
Console ×
<terminated> App [Java Application] C:\Users\Raj\.p2\pool\plugins\c
Song Started
Song Ended
Yahoo Yahoo : After Returing Aspect
```

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

AspectClass

```
Package aop;

import org.aspectj.lang.annotation.After;
import
org.aspectj.lang.annotation.AfterReturning;
 import
org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Around;
import
org.aspectj.lang.annotation.Aspect;
import
org.aspectj.lang.annotation.Before;
import
org.aspectj.lang.annotation.Pointcut;
@Aspect
public class mcaAspect{


      @Pointcut("execution(*brook.makeSong(..))")
    Private void selectAll(){}

    @AfterThrowing(pointcut="selectAll()",throwing="error")
    public void afterThrowingAdvice(IllegalArgumentException error){
        System.out.println("YahooYahoo:Therehasbeenan exception:");
    }

}
```

TargetClass

```
Package aop;

Public class brook implements Guitar{


      public void makeSong(){

            System.out.println("Song
            Started");
            System.out.println("Song Ended");
        Throw new IllegalArgumentException("An error occurredwhile
making the song.");

      }


}
```

Output :

```
<terminated> App [Java Application] C:\Users\Raj\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre\bin\javaw.exe  (25 Dec 2023, 14:12:23 – 14:
Song Started
Song Ended
Yahoo Yahoo : There has been an exception:
Exception in thread "main" java.lang.IllegalArgumentException: An error occurred while making the song.
        at aop.brook.makeSong(brook.java:10)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)
        at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:568)
        at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:344)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.invokeJoinpoint(ReflectiveMethodInvocation.java:198)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:163)
        at org.springframework.aop.aspectj.AspectJAfterThrowingAdvice.invoke(AspectJAfterThrowingAdvice.java:62)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
        at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:95)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:186)
        at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:212)
        at jdk.proxy2/jdk.proxy2.$Proxy10.makeSong(Unknown Source)
```

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

AspectClass

```
Package aop;

import org.aspectj.lang.annotation.After;
import
org.aspectj.lang.annotation.AfterReturning;
import
org.aspectj.lang.annotation.AfterThrowing;
import org.aspectj.lang.annotation.Around;
import
org.aspectj.lang.annotation.Aspect;
import
org.aspectj.lang.annotation.Before;
import
org.aspectj.lang.annotation.Pointcut;

@Aspect
Public class mcaAspect{


        @Pointcut("execution(*brook.makeSong())")
        Public void songPointCut(){
                System.out.println("YahooYahoo:Iampointcut");
        }

        @AfterReturning("songPointCut()")
      Public void afterSong(){
          System.out.println("YahooYahoo:UsedBYPointcut");
      }
}
```

Output:

```
Console ×
<terminated> App [Java Application] C:\Users\Raj\.p2\pool
Song Started
Song Ended
Yahoo Yahoo : Used BY Pointcut
```

Practical Assignment No 9

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

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/4.0.0http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mca</groupId>
  <artifactId>springJDBC</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>springJDBC</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>

      <!--https://mvnrepository.com/artifact/org.springframework/spring-core-->
      <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>5.2.3.RELEASE</version>
      </dependency>

        <!--https://mvnrepository.com/artifact/org.springframework/spring-context-->
      <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-context</artifactId>
          <version>5.2.3.RELEASE</version>
      </dependency>

      <!--https://mvnrepository.com/artifact/org.springframework/spring-jdbc-->
      <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-jdbc</artifactId>
          <version>5.2.3.RELEASE</version>
      </dependency>

       <!--https://mvnrepository.com/artifact/mysql/mysql-connector-java-->
      <dependency>
          <groupId>mysql</groupId>
          <artifactId>mysql-connector-java</artifactId>
          <version>8.0.20</version>
      </dependency>

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

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
"xmlns:p="http://www.springframework.org/schema/p"xmlns:c="
http://www.springframework.org/schema/c"xsi:schemaLocation="
http://www.springframework.org/schema/beanshttp://www.springfr
amework.org/schema/beans/spring-
beans.xsdhttp://www.springframework.org/schema/contexthttp://
www.springframework.org/schema/context/spring- context.xsd">


<beanclass="org.springframework.jdbc.datasource.DriverManagerDataSource"name="ds">
<propertyname="driverClassName"value="com.mysql.jdbc.Driver"/>
<propertyname="url"value="jdbc:mysql://localhost:3306/springjdbc"/>
<propertyname="username"value="root"/>
<propertyname="password"value="root"/>
</bean>

<beanclass="org.springframework.jdbc.core.JdbcTemplate"name="jdbcTemplate"p:dataSource-ref="ds"/>

</beans>
```

MainClass

```
Package com.mca;

Import org.springframework.context.ApplicationContext;
Import org.springframework.context.support.ClassPathXmlApplicationContext;
Import org.springframework.jdbc.core.JdbcTemplate;

Public class App
{
	Public static void main(String[]args)
	{
		System.out.println("kaizokuoniorewanaru!");
		ApplicationContext context= new ClassPathXmlApplicationContext("com/mca/config.xml");
		JdbcTemplate temp =context.getBean("jdbcTemplate",JdbcTemplate.class);

//insertQuery
		String query1="insert into strawHat values(?,?,?)";
		String query2="update strawHat set bounty=? whereid=?";
		 String query3 = "delete from strawHat whereid=?";

//firequery
		Int result1=temp.update(query1,2,"zoro","1.2Billion");
		System.out.println("Number ofr ecords insetred "+ result1);

		Int result2=temp.update(query2,"4billion",1);

		System.out.println("Numberofrecordsupdated "+ result2);

		Int result3=temp.update(query3,5);

		System.out.println("Number of records Deleted "+ result3);
	}
}
```

Output:

Result Grid | Filter Rows: | Edit:

id	name	bounty
1	luffy	3 Billion
5	dummy	entry
NULL	NULL	NULL

Console ×

```
<terminated> App (1) [Java Application] C:\Users\Raj\.p2\pc
kaizokuo ni ore wa naru!
Loading class `com.mysql.jdbc.Driver'. This is dep
Number of records insetred 1
Number of records updated 1
Number of records Deleted 1
```

Result Grid | Filter Rows:

id	name	bounty
1	luffy	4 billion
2	zoro	1.2 Billion
NULL	NULL	NULL

b. Write a program to demonstrate Prepared Statement in Spring JDBC Template.

MainClass

```
Package com.mca;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCreator;

public class App
{
    Public static void main(String[] args)
    {
        System.out.println("kaizokuoniorewanaru!");
        ApplicationContext context= new ClassPathXmlApplicationContext("com/mca/config.xml");
        JdbcTemplate temp = context.getBean("jdbcTemplate", JdbcTemplate.class);

        String query1="insert into strawHat(id,name,bounty)values(?,?,?)";
        Int result=temp.update(new PreparedStatementCreator(){

                        @Override
                        Public PreparedStatementcreatePreparedStatement(Connectioncon)throws
SQLException{

                                PreparedStatement ps=con.prepareStatement(query1); ps.setInt(1,
                                3);
                                ps.setString(2, "zoro");
                                ps.setString(3,"1.1Billion");

                                return ps;
                        }
                });
```

Output:

Result Grid | Filter Rows:

id	name	bounty
1	luffy	4 billion
2	zoro	1.2 Billion
NULL	NULL	NULL

Console ×

```
<terminated> App (1) [Java Application] C:\Users\Raj\.p2
kaizokuo ni ore wa naru!
Loading class `com.mysql.jdbc.Driver'. This is
Number of rows afftected 1
```

Result Grid | Filter Rows:

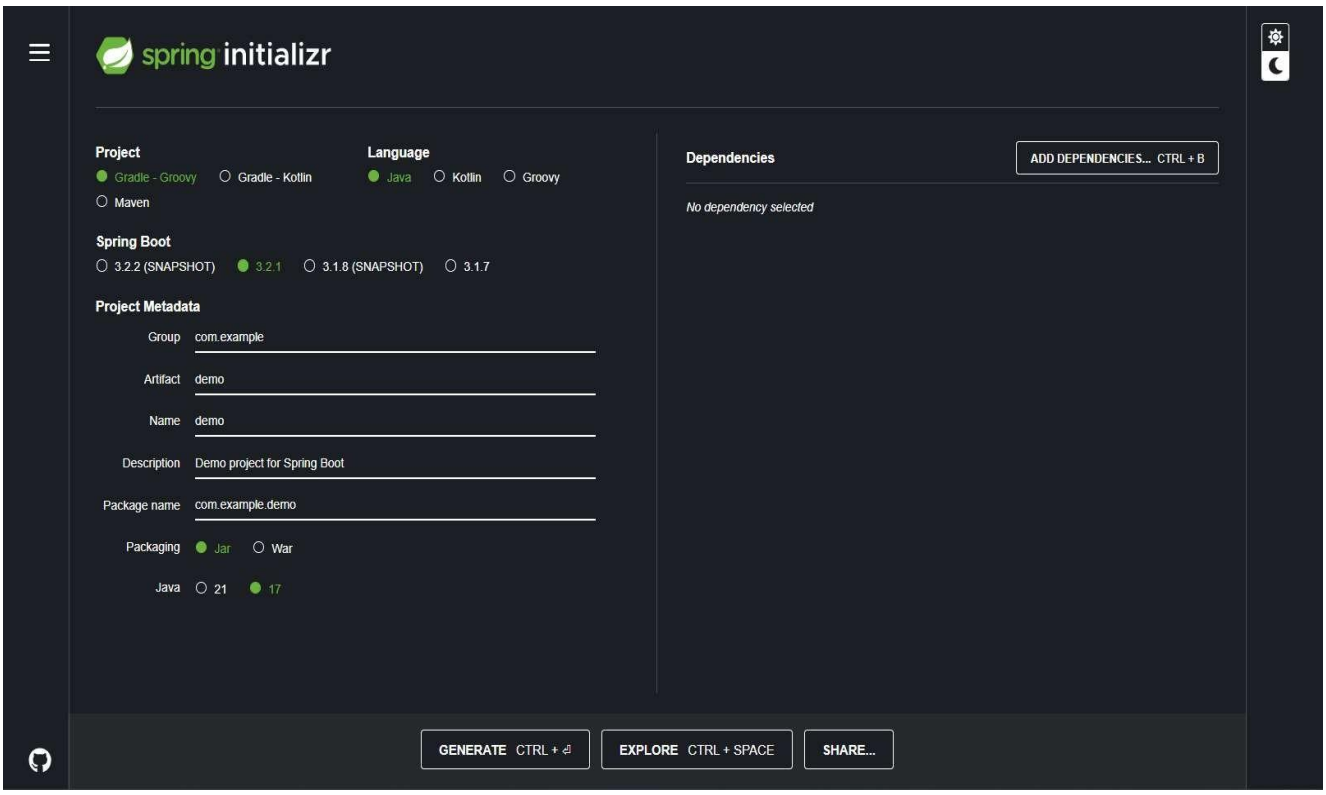
id	name	bounty
1	luffy	4 billion
2	zoro	1.2 Billion
3	zoro	1.1 Billion
NULL	NULL	NULL

Practical Assignment No 10

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

Step1:

Go to Spring Initializr. Select the type of project (Maven).
Choose the language (Java).
Select the Spring Boot version.
Fillin the project metadata. Add the necessary dependencies (at least spring-boot- starter-web). Click on "Generate" to download the project.



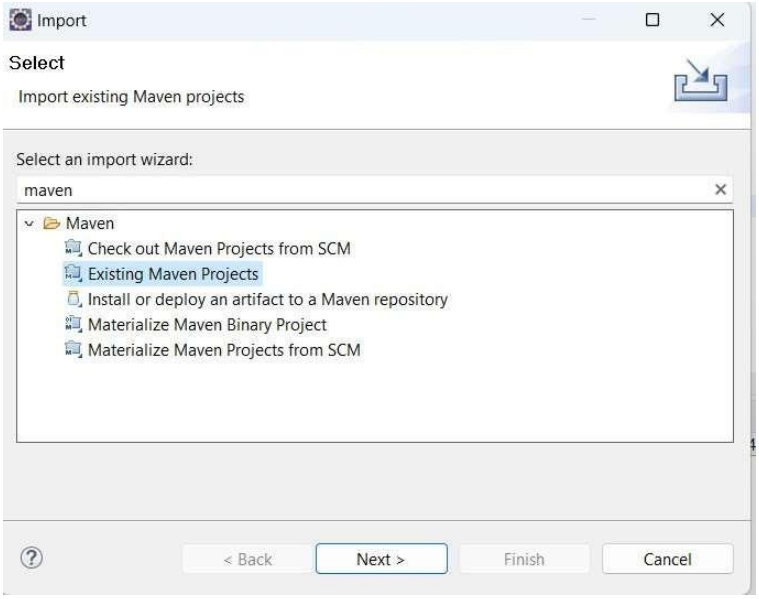
Step2:

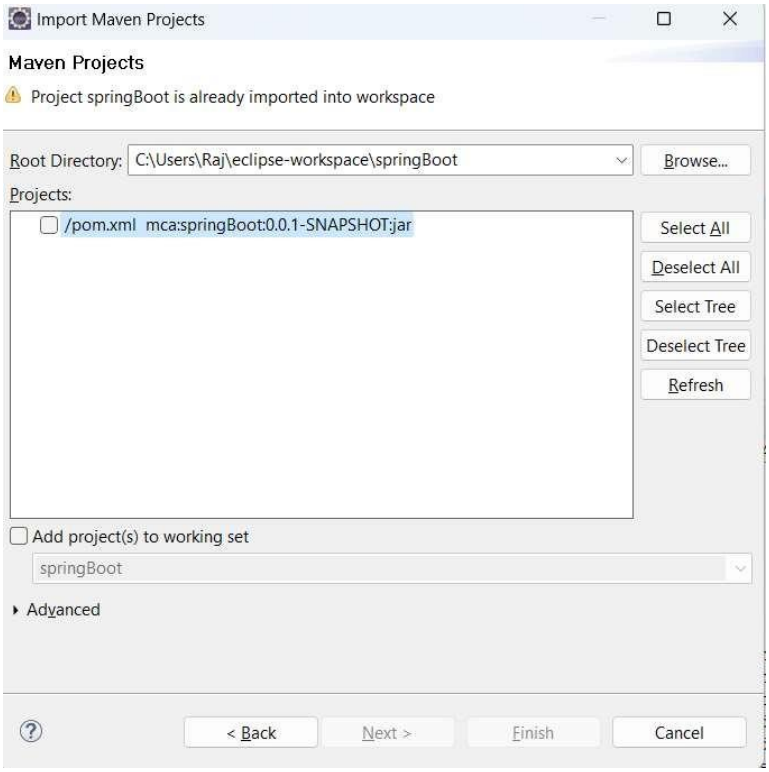
Open Eclipse IDE. Navigate to File > Import.

Select "Existing Maven Projects".

Click on "Next". Click on "Browse" and navigate to the location where you downloaded the project.

Make sure the pom.xml file is checked. Click on "Finish".


Import
Select
Import existing Maven projects
Select an import wizard:
maven
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
< Back
Next >
Finish
Cancel


Import Maven Projects
Maven Projects
Project springBoot is already imported into workspace
Root Directory:
C:\Users\Raj\eclipse-workspace\springBoot
Browse...
Projects:
/pom.xml mca:springBoot:0.0.1-SNAPSHOT:jar
Select All
Deselect All
Select Tree
Deselect Tree
Refresh
Add project(s) to working set
springBoot
Advanced
< Back
Next >
Finish
Cancel

MainClass

```
Package com.mca.spring;


Import org.springframework.boot.SpringApplication;
Import org.springframework.boot.autoconfigure.SpringBootApplication;
Import org.springframework.web.bind.annotation.GetMapping;
Import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
Public class myApplication{


    Public static void  (String[] args){SpringApplication.run(myApplication.class,args);

    }


    @RestController
    public class controller {
        @GetMapping("/")
        public String quote(){

            return "Hero?No!We’repirates!Iloveheroes,butIdon’twanna
beone!";
        }

    }

}
```

Output:

Hero? No! We’re pirates! I love heroes, but I don’t wanna be one!