

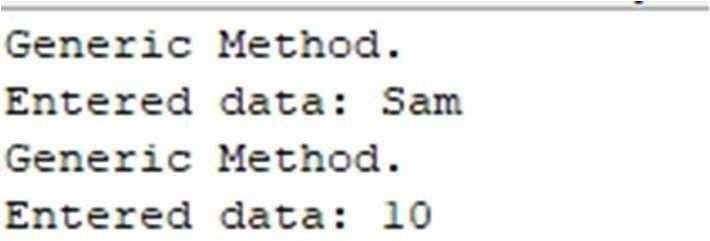
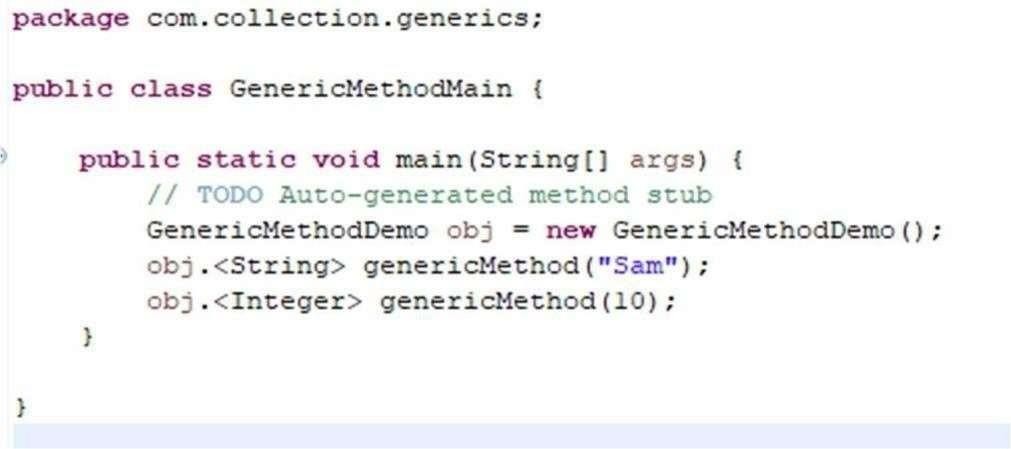
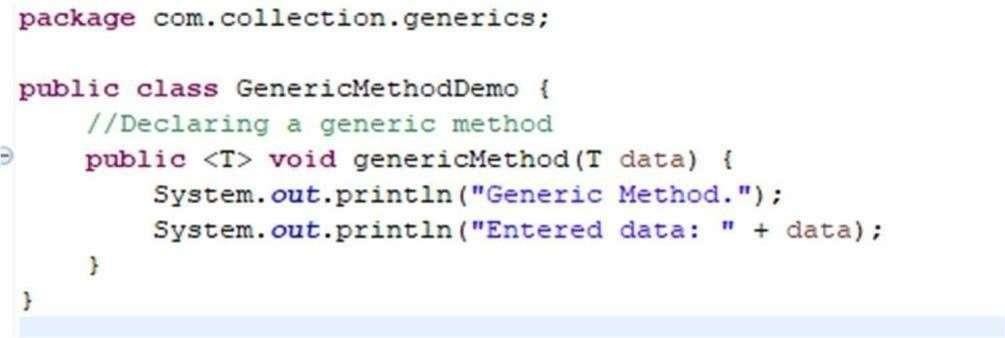
#### PRACTICAL NO. 1

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

**Generic Class :**

**Main Class :**

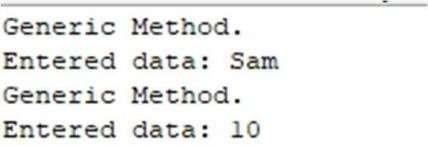
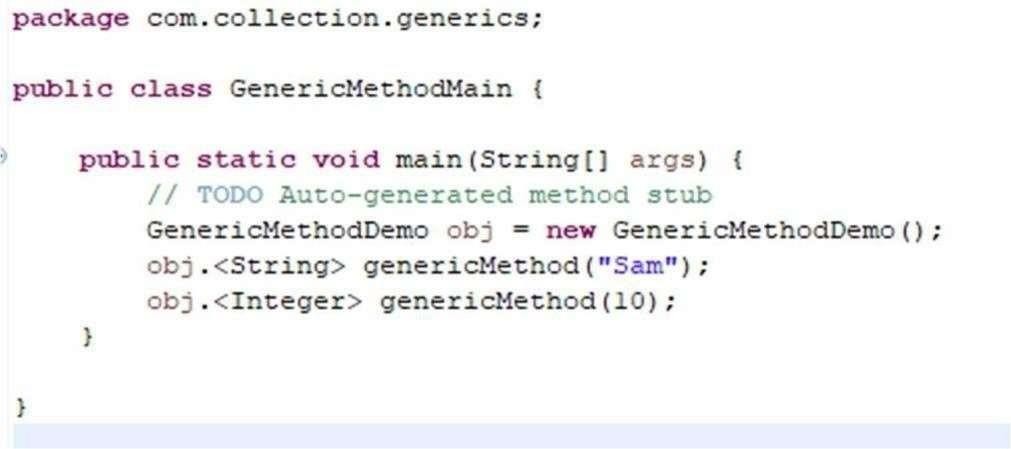
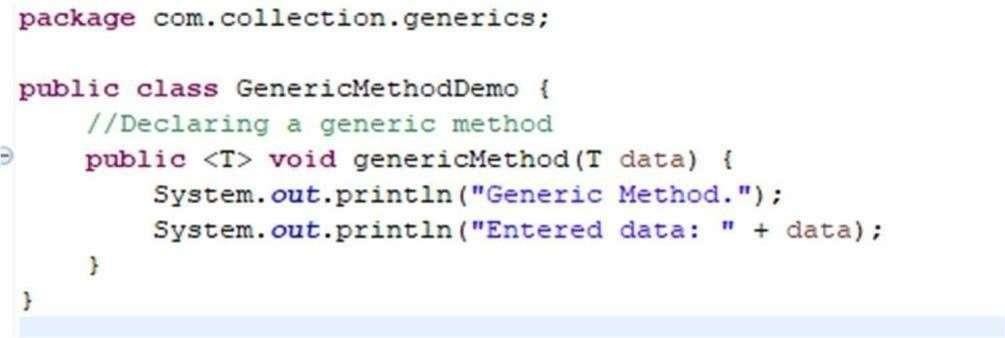
#### OUTPUT:



1. **Aim: Write a Java Program to demonstrate Generic Methods. Generic Method Class**:

**Main Class:**

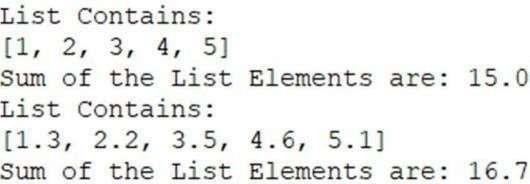
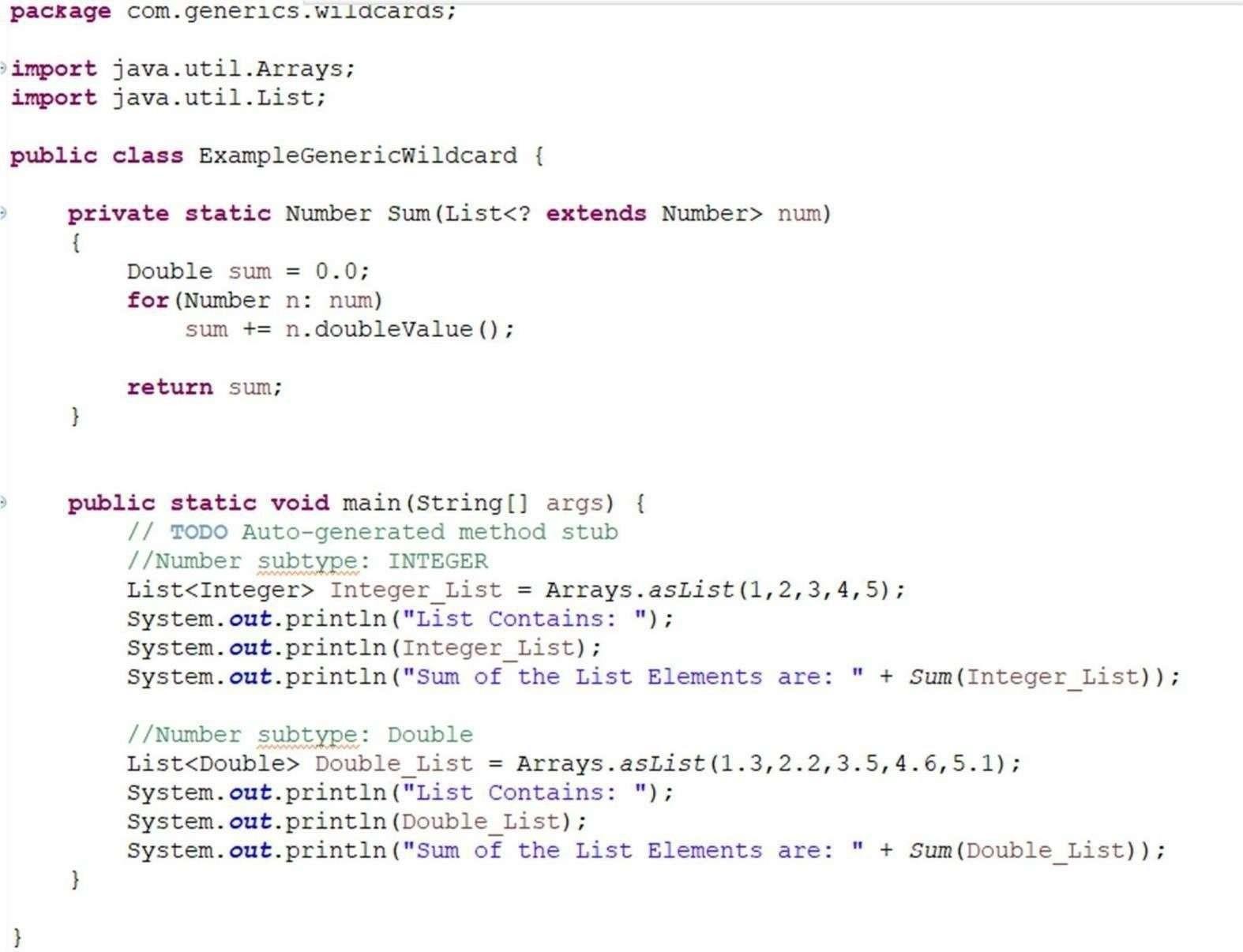
#### OUTPUT :



1. **Aim: Write a Java Program to demonstrate Generic Methods. Generic Method Class:**

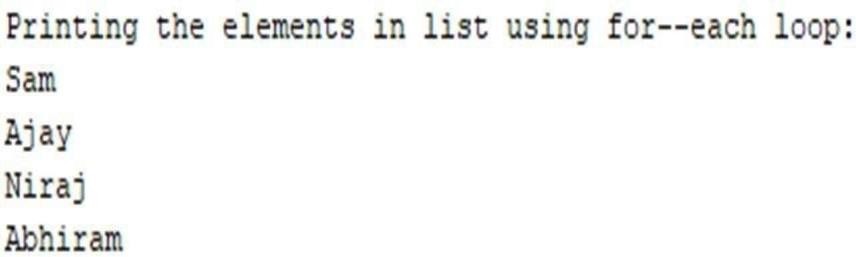
## Main Class:

#### OUTPUT :



1. **Aim: Write a Java Program to demonstrate Wildcards in Java Generics.**

#### OUTPUT :



**PRACTICAL NO. 2**

**[A] Aim: 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.**

#### [B]

package TestJavaCollection; import java.util.\*;

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

ArrayList<String> al = new ArrayList<String>(); al.add("Sam");

al.add("Ajay");

al.add("Niraj");

al.add("Abhiram");

System.out.println("Printing the elements in list using for-each loop:"); for (String var : al) {

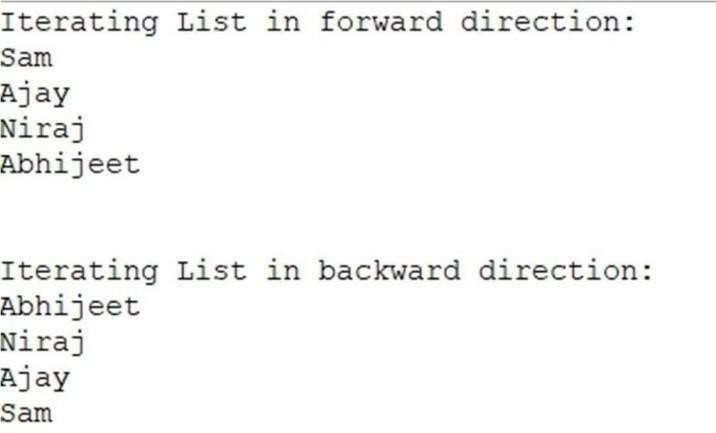
System.out.println(var);

}

}

}

#### OUTPUT :



**[C] Aim: 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) {

ArrayList<String> al = new ArrayList<String>(); al.add("Sam");

al.add("Ajay");

al.add("Niraj");

al.add("Abhijeet");

System.out.println("Iterating List in forward direction:"); ListIterator<String> liForward = al.listIterator();

while (liForward.hasNext()) { System.out.println(liForward.next());

}

System.out.println("\n");

System.out.println("Iterating List in backward direction:"); ListIterator<String> liBack = al.listIterator(al.size()); while (liBack.hasPrevious()) {

System.out.println(liBack.previous());

}

}

}

#### OUTPUT :

**PRACTICAL NO. 3**

**[A]Aim: Write a Java program to create a Set containing list of items of type String and print the items in the list using Iterator interface. Also print the list in reverse /backward direction.**

**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>()**

**// Add items to the set obj.add("Sam");**

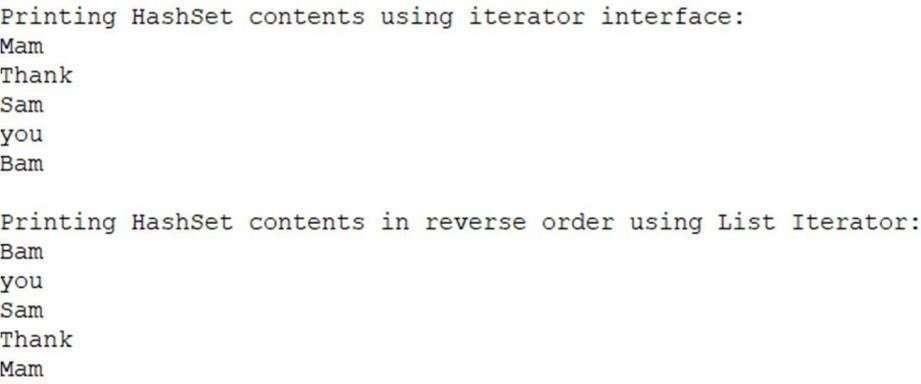
**obj.add("Bam");**

**obj.add("Thank");**

**obj.add("you");**

**obj.add("Mam");**

**System.out.println("Printing HashSet contents using iterator interface:"); Iterator<String> itr = obj.iterator()**



**while (itr.hasNext()) {**

**System.out.println(itr.next());**

**}**

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

**// Convert the set to a List**

**List<String> setToList = new ArrayList<>(obj); System.out.println("Printing HashSet contents in reverse order using**

**ListIterator:");**

**// Print the list in reverse order**

**ListIterator<String> li = setToList.listIterator(setToList.size());**

**while (li.hasPrevious()) { System.out.println(li.previous());**

**}**

**}**

## Output:

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

package com.collection.set.example; import java.util.HashSet;

import java.util.Iterator;

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

HashSet<String> obj = new HashSet<String>();

// Add items to the set obj.add("Sam");

obj.add("Bam");

// Display the contents of the set System.out.println("Set Contains: "); Iterator<String> itr = obj.iterator(); while (itr.hasNext()) {

System.out.println(itr.next());

}

// Insert items of one set into another set System.out.println("\nCopying contents of one set into another:"); HashSet<String> obj1 = new HashSet<String>(); obj1.add("Thank");

obj1.add("you");

obj1.add("Mam");

obj1.addAll(obj); // Add all elements of obj to obj1 System.out.println("New Set Contains:"); Iterator<String> it = obj1.iterator();

while (it.hasNext()) { System.out.println(it.next());

}

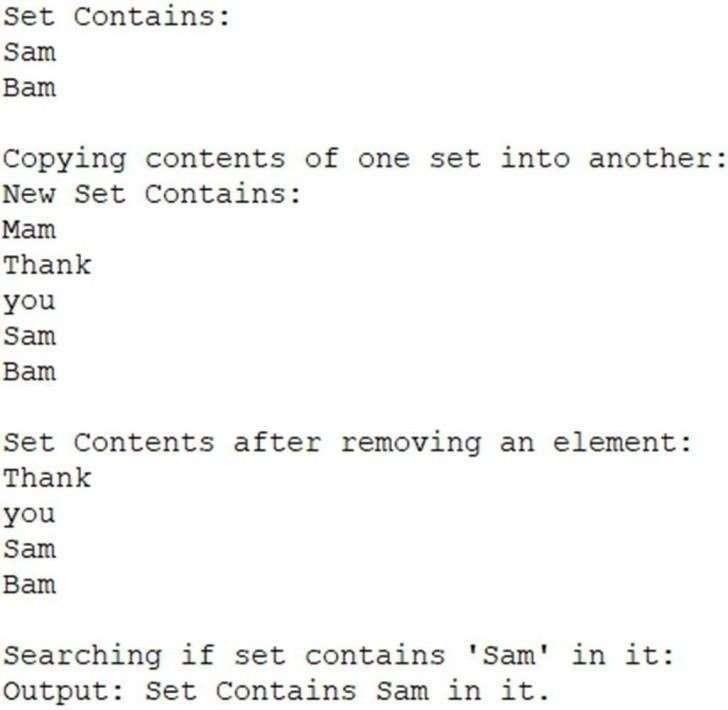
// Remove an item from the set System.out.println("\nRemoving an item from the set:"); obj1.remove("Mam");

System.out.println("Set Contents after removing an element:"); Iterator<String> it1 = obj1.iterator();

while (it1.hasNext()) { System.out.println(it1.next());

}

// Search for a specified item in the set



System.out.println("\nSearching if set contains 'Sam' in it:"); boolean value = obj.contains("Sam");

if (value) {

System.out.println("Output: Set contains 'Sam' in it.");

} else {

System.out.println("Output: Set does not contain 'Sam' in it.");

}

}

}

#### OUTPUT :

**PRACTICAL NO. 4**

1. **Aim: 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) {

Map<Integer, String> m = new HashMap<Integer, String>();

// Add items to the map m.put(1, "Sam");

m.put(2, "Bam"); System.out.println("Map contains: " + m);

// Get value of a specified key System.out.println("\nGetting value of specified key:"); String value = m.get(1);

System.out.println("Value stored at key 1: " + value);

// Insert elements of one map into another map System.out.println("\nInserting elements of one map into another map:"); Map<Integer, String> m1 = new HashMap<Integer, String>(); m1.put(1, "Thank");

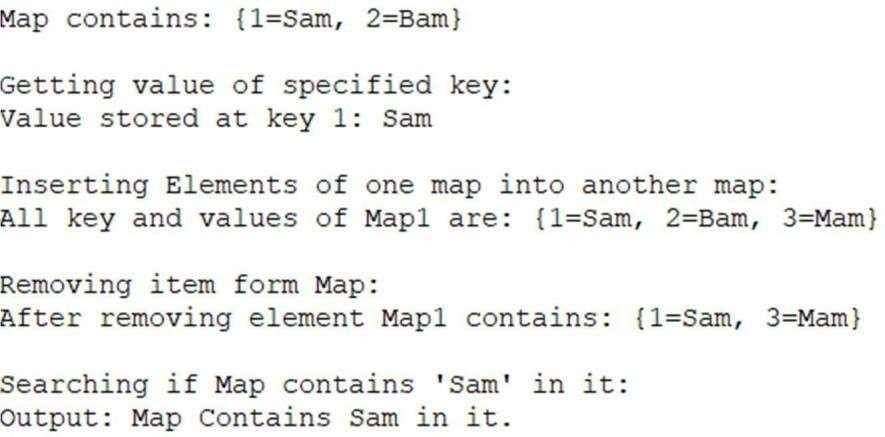
m1.put(2, "you");

m1.put(3, "Mam");

m1.putAll(m); // Insert all elements from m to m1 System.out.println("All keys and values of Map1 are: " + m1);

// Remove items from map System.out.println("\nRemoving item from Map:"); m1.remove(2); // Remove key 2

System.out.println("After removing element, Map1 contains: " + m1);



// Search for a specified key in the map System.out.println("\nSearching if Map contains 'Sam' in it:"); boolean value1 = m1.containsValue("Sam");

if (value1) {

System.out.println("Output: Map contains 'Sam' in it.");

} else {

System.out.println("Output: Map does not contain 'Sam' in it.");

}

}

}

# OUTPUT :



#### PRACTICAL NO. 5

1. **Aim: Write a Java program using Lambda Expression to print “Hello, India”.**

package com.example.assignment; @FunctionalInterface

public interface Q2LambdaHelloIndia {

// Declaring an abstract method void printHelloIndia();

}

package com.example.assignment; public class Q2Lambda {

public static void main(String[] args) {

Q2LambdaHelloIndia obj = () -> System.out.println("Hello, India!"); obj.printHelloIndia();

}

}

#### OUTPUT :



1. **Aim: Write a Java program using Lambda Expression with single parameters.**

package com.lambda.example; @FunctionalInterface

interface LambdaExampleInterface { void print(String message);

}

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

LambdaExampleInterface printMessage = (message) -> System.out.println(message); printMessage.print("Hello India!");

}

}

#### OUTPUT :

1. **Aim : Write a Java program using Lambda Expression with multiple parameters to add and multiply two numbers.**

**File 1:**

package com.example.assignment; @FunctionalInterface

public interface Q4AddLambda {

void add(int num1, int num2);

}

**File 2:**

package com.example.assignment; @FunctionalInterface

public interface Q4MultiplyLambda {

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 (Scanner s = new Scanner(System.in)) { System.out.println("Enter First Number: "); int num1 = s.nextInt(); System.out.println("Enter Second Number: "); int num2 = s.nextInt();

Q4AddLambda obj1 = (value1, value2) -> {

System.out.println("Addition of given numbers is: " + (value1 + value2));

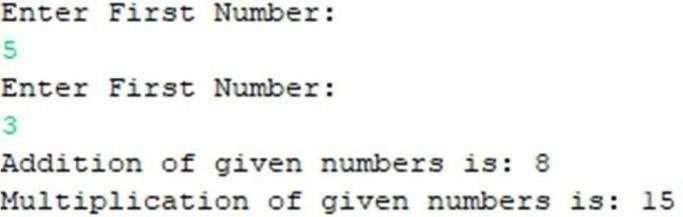
};

Q4MultiplyLambda obj2 = (value1, value2) -> { System.out.println("Multiplication of given numbers is: " + (value1 \* value2));

};

// Calling the lambda methods obj1.add(num1, num2); obj2.multiply(num1, num2);

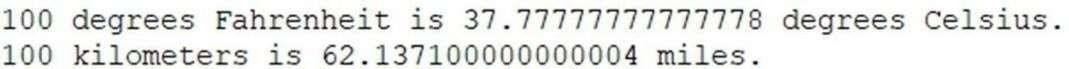
}



}

}

#### OUTPUT :



**[D] Aim: Write a Java program using Lambda Expression to calculate the following:**

* **Convert Fahrenheit to Celsius**
* **Convert Kilometres to Miles.**

package com.lambda.example; @FunctionalInterface

interface Convert {

double convertInput(double input);

}

public class LambdaConvert {

public static void main(String[] args) {

Convert toCelsius = (fahrenheit) -> (fahrenheit - 32) \* 5.0 / 9.0;

Convert toMiles = (kilometers) -> kilometers \* 0.621371; double celsius = toCelsius.convertInput(100);

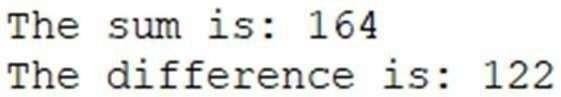
System.out.println("100 degrees Fahrenheit is " + celsius + " degrees Celsius.");

double miles = toMiles.convertInput(100); System.out.println("100 kilometers is " + miles + " miles.");

}

}

# OUTPUT:



1. **Aim: Write a Java program using Lambda Expression with or with out 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; Subtract subtracting = (a, b) -> a - b;

int sum = adding.add(155, 9); System.out.println("The sum is: " + sum); int difference = subtracting.subtract(124, 2);

System.out.println("The difference is: " + difference);

}

}

#### OUTPUT :



1. **Aim: Write a Java program using Lambda Expression to concatenate two strings.**

package com.lambda.example; @FunctionalInterface

interface ConcatenateInterface {

String concatenate(String a, String b);

}

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

ConcatenateInterface concat = (a, b) -> a + b; String str1 = "Hello,";

String str2 = "India!";

String result = concat.concatenate(str1, str2); System.out.println(result);

}

}

#### OUTPUT :



**PRACTICAL NO. 6**

**[A] Aim: Write Program to demonstrate different Implicit Objects**

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

<%@ 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>Out Object</h1>

<%

out.println("Luffy: This is... a love or deal");

%>

<h1>Request Object</h1>

<%

String uri = request.getRequestURI(); out.println("Requested URI: " + uri);

%>

<h1>Session Object</h1>

<%

session.setAttribute("luffy", "I refuse your refusal"); String attribute = (String) session.getAttribute("luffy");

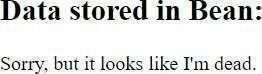
out.println("The value of the session attribute 'luffy' is: " + attribute);

%>

</body>

</html>

**OUTPUT :**



## Aim: Write Program to demonstrate temporary storage using Bean.

<%@ page import="java.util.ArrayList" %>

<jsp:useBean id="myBean" class="jspExample.MyBean" scope="request" />

<%

// Set data in the bean

myBean.setData("Sorry, but it looks like I'm dead.");

// Retrieve data from the bean String data = myBean.getData();

%>

<html>

<head><title>Temporary Storage Using Bean</title></head>

<body>

<h2>Data stored in Bean:</h2>

<p><%= data %></p>

</body>

</html>

#### OUTPUT :



**[C] Aim: Write a program to demonstrate Standard Action tags.**

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

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Practical 7</title>

</head>

<body>

<%@ include file="header.jsp" %> <!-- Directive to include header -->

<%-- JSP Declaration --%>

<%! int count = 0; %>

<%-- JSP Scriptlet --%>

<%

count++;

out.println("This is an Example of scriptlet. Count is now: " + count);

%>

<%-- JSP Expression --%>

<p>This is an Example of Directive expression. The value of count is now: <%= count

%></p>

<%@ include file="footer.jsp" %> <!-- Directive to include footer -->

</body>

</html>

# OUTPUT:



**[D] Aim: 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>" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>JSP Directives</title>

</head>

<body>

<h2>Welcome to JSP Directives!</h2>

<c:out value="${'I still have my friends!'}" />

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

</body>

</html>

**OUTPUT:**



## [E] Aim: Write a program to demonstrate Session Tracking using Cookies.

<%@ page import="java.io.PrintWriter" %>

<%

// Get the current session or create a new one HttpSession session1 = request.getSession(true);

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

// Create a cookie for the username

Cookie usernameCookie = 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 :



**[F] Aim: 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>JSTL Demo</title>

</head>

<body>

<h2>JSTL Core Tags Demo</h2>

<c:set var="message" value="I love heroes, but I don't want to be one." />

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

<c:if test="${5 > 3}">

<p>The condition is true.</p>

</c:if>

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

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

</c:forEach>

</body>

</html>

# OUTPUT :

**[G] Aim: 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" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Practical 1</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: 1px solid #ddd; padding: 8px;

}

th {

padding-top: 12px; padding- bottom: 12px; text-align: left; background-color: purple; color: white;

}

input {

height: 20px; padding: 5px 10px;

}

</style>

</head>

<body>

<h1>Add a new entry</h1>

<form method="get">

<label for="search">Search:</label>

<input type="text" id="search" name="search" placeholder="Search by 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 ?;

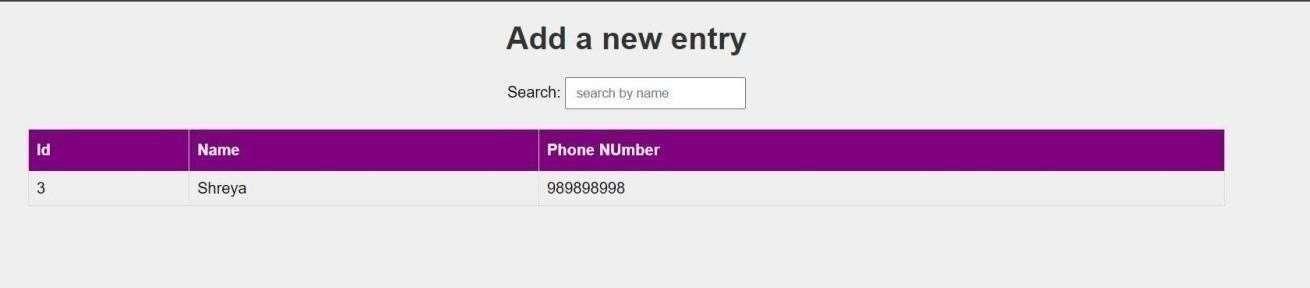
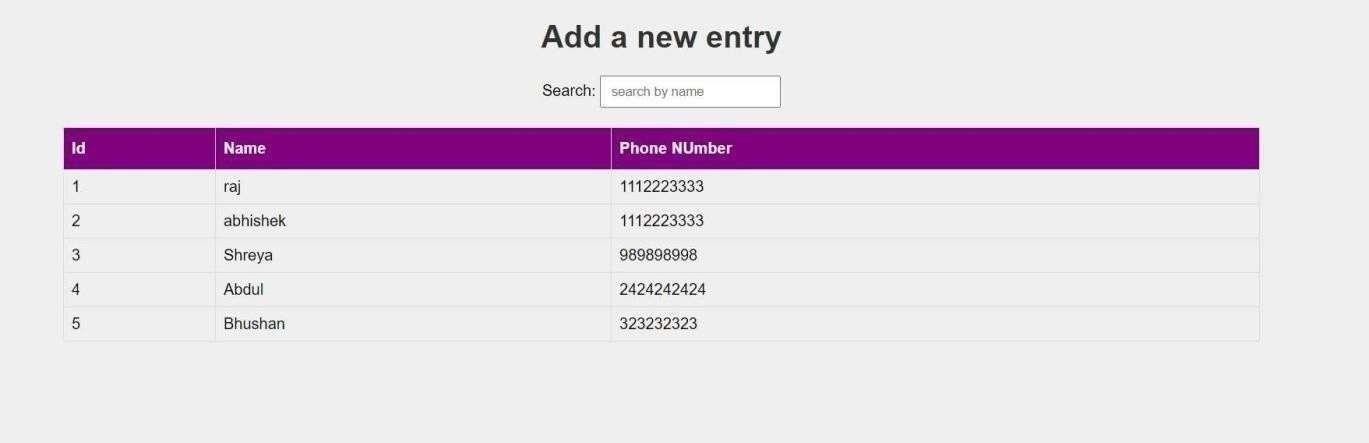
<sql:param value="%${param.search}%" />

</sql:query>

<table border="1" width="100%">

<tr>

<th>Id</th>



<th>Name</th>

<th>Phone Number</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>

</html>

# OUTPUT :

**[H] Aim: 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" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="ISO-8859-1">

<title>Practical 2</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: 1px solid black; border-radius: 4px;

}

input[type=text], input[type=password] { width: 100%;

padding: 12px 20px; margin: 8px 0; display: inline-block; border: 1px solid #ccc; box-sizing: border-box;

}

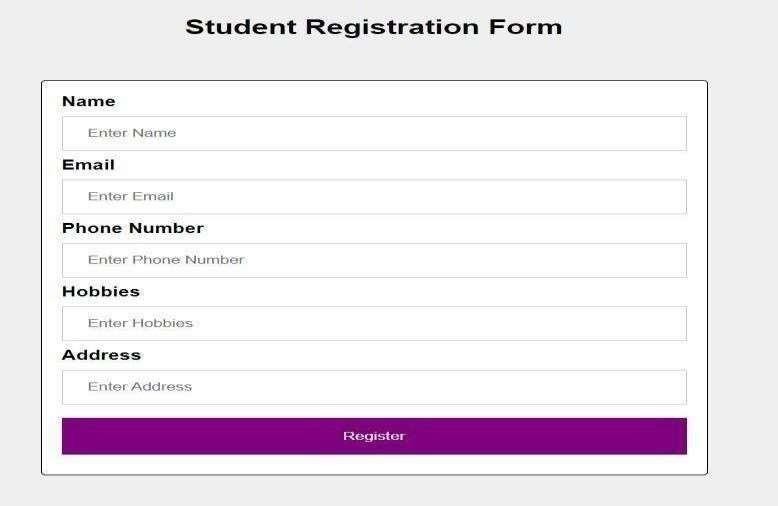
button {

background-color: purple; color: white;

padding: 14px 20px; margin: 8px 0; border: none; cursor: pointer; width: 100%;

}

button:hover {



opacity: 0.8;

}

h2 {

text-align: center;

}

</style>

</head>

<body>

<h2>Student Registration Form</h2>

<div class="container">

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

<label for="name"><b>Name</b></label>

<input type="text" placeholder="Enter Name" name="name" required>

<label for="email"><b>Email</b></label>

<input type="text" placeholder="Enter Email" name="email" required>

<label for="phone"><b>Phone Number</b></label>

<input type="text" placeholder="Enter Phone Number" name="phone" required>

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

<input type="text" placeholder="Enter Hobbies" name="hobbies" required>

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

<input type="text" placeholder="Enter Address" name="address" required>

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

</form>

</div>

</body>

</html>

#### OUTPUT :

**PRACTICAL NO. 7**

1. **Aim : 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() { return name;

}

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);

}

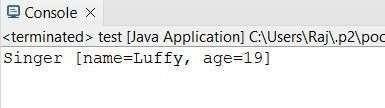
}

**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/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>">



<bean id="Singer" class="com.example.SpringTest.Singer">

<property name="name" value="Luffy" />

<property name="age" value="19" />

</bean>

</beans>

**SingerTest.java**

package com.example.SpringTest;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class SingerTest {

public static void main(String[] args) { ApplicationContext context = new

ClassPathXmlApplicationContext("ApplicationContext.xml");

Singer singer = (Singer) context.getBean("Singer"); singer.displayInfo();

}

}

#### OUTPUT :

1. **Aim: 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;

// Getter and Setter methods public String getName() {

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 with parameters

public Zoro(String name, double height, int swords) { super();

this.name = name; this.height = height; this.swords = swords;

}

// Default constructor public Zoro() {

super();

}

// toString method @Override

public String toString() {

return "Name of Character = " + name + ", Height of Character = " + height + ", No. of swords = " + swords;

}

}

**ApplicationContext.xml**

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

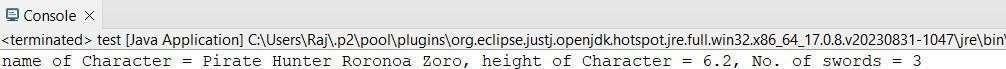
<beans xmlns="<http://www.springframework.org/schema/beans>" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>" xmlns:context="<http://www.springframework.org/schema/context>" xmlns:p="<http://www.springframework.org/schema/p>" xsi:schemaLocation="<http://www.springframework.org/schema/beans>

<http://www.springframework.org/schema/beans/spring->

beans.xsd

<http://www.springframework.org/schema/context> <http://www.springframework.org/schema/context/spring->

context.xsd">



<bean id="zoro" class="MCA.Zoro" p:name="Pirate Hunter Roronoa Zoro" p:height="6.2" p:swords="3" />

</beans> **Main Class** package MCA;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

public static void main(String[] args) { ApplicationContext context = new

ClassPathXmlApplicationContext("MCA/mcaConfig.xml");

Zoro temp = (Zoro) context.getBean("zoro"); System.out.println(temp);

}

}

**Maven Dependencies:**

#### OUTPUT :

1. **Aim: Write a program to demonstrate dependency injection via Constructor. (Primitive)**

**Pojo Class**

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 String toString() {

return "Character name=" + name + ", No. of gears=" + 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/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>">

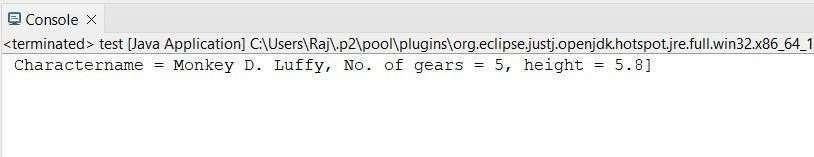
<!-- Bean definition with constructor injection -->

<bean class="MCA.Luffy" name="luffy">

<constructor-arg value="Monkey D. Luffy"/>

<constructor-arg value="5"/>

<constructor-arg value="5.8"/>



</bean>

</beans>

**Main Class**

package MCA;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

public static void main(String[] args) { ApplicationContext context = new

ClassPathXmlApplicationContext("MCA/applicationContext.xml");

Luffy temp = (Luffy) context.getBean("luffy");

System.out.println(temp);

}

}

#### OUTPUT :

1. **Aim : Write a program to demonstrate dependency injection via setter method. (Non-Primitive)**

**Pojo Class**

package MCA; public class Sanji {

private String name;

private double height;

private Zoro obj; // Reference to another object (Zoro)

// Getters and Setters public String getName() {

return name;

}

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

}

public double getHeight() { return height;

}

public void setHeight(double height) { this.height = height;

}

public Zoro getObj() { return obj;

}

public void setObj(Zoro obj) { this.obj = obj;

}

// Constructor for Dependency Injection

public Sanji(String name, double height, Zoro obj) { super();

this.name = name; this.height = height; this.obj = obj;

}

// Default constructor

public Sanji() { super();

}

// toString method @Override

public String toString() {

return "Sanji [name=" + name + ", height=" + height + ", obj=" + obj + "]";

}

}

**Reference Class**

package MCA; public class Zoro {

private String name;

private double height; private int swords;

// Getters and Setters public String getName() {

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;

}

public Zoro(String name, double height, int swords) { super();

this.name = name; this.height = height; this.swords = swords;

}

public Zoro() { super();

}

@Override

public String toString() {

return "Zoro [name=" + name + ", height=" + height + ", swords=" + 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/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>">

<!-- Bean for Zoro -->

<bean class="MCA.Zoro" name="zoro">

<constructor-arg value="Pirate Hunter Roronoa Zoro" />

<constructor-arg value="6.2" />

<constructor-arg value="3" />

</bean>

<!-- Bean for Sanji, with reference injection to Zoro -->

<bean class="MCA.Sanji" name="sanji">

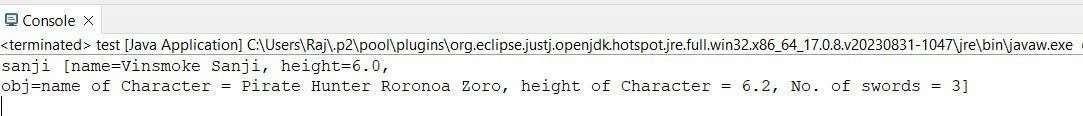
<constructor-arg value="Vinsmoke Sanji" />

<constructor-arg value="6.0" />

<constructor-arg ref="zoro" />

</bean>

</beans>



**Main Class**

package MCA;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

public static void main(String[] args) { ApplicationContext context = new

ClassPathXmlApplicationContext("MCA/applicationContext.xml");

Sanji temp = (Sanji) context.getBean("sanji"); System.out.println(temp);

}

}

#### OUTPUT:

1. **Aim : Write a program to demonstrate dependency injection via Constructor. (Non- Primitive) By Ref**

**Pojo Class**

package MCA; public class Ussop {

private String name;

private double height;

private Luffy obj; // Reference to another object (Luffy)

// Constructor for Dependency Injection

public Ussop(String name, double height, Luffy obj) { super();

this.name = name; this.height = height; this.obj = obj;

}

// toString method @Override

public String toString() {

return "Ussop [Name=" + name + ", height=" + height + ", obj=" + obj + "]";

}

}

**Reference Class**

package MCA; public class Luffy {

private String name;

private int gears; private double height;

// Constructor for Dependency Injection

public Luffy(String name, int gears, double height) { super();

this.name = name; this.gears = gears; this.height = height;

}

// toString method

@Override

public String toString() {

return "Character name=" + name + ", No. of gears=" + 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/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>">

<!-- Bean for Luffy -->

<bean class="MCA.Luffy" name="luffy">

<constructor-arg value="Monkey D. Luffy"/>

<constructor-arg value="5"/>

<constructor-arg value="5.8"/>

</bean>

<!-- Bean for Ussop, with reference injection to Luffy -->

<bean class="MCA.Ussop" name="ussop">

<constructor-arg value="Sogeking Ussop"/>

<constructor-arg value="5.11"/>

<constructor-arg ref="luffy"/>

</bean>

</beans>

**Main Class**

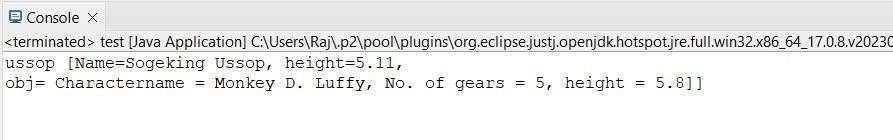
package MCA;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

public static void main(String[] args) {



ApplicationContext context = new ClassPathXmlApplicationContext("MCA/applicationContext.xml");

Ussop temp = (Ussop) context.getBean("ussop"); System.out.println(temp);

}

}

#### OUTPUT :

1. **Aim : Write a program to demonstrate dependency injection via Constructor. (Collection)**

**Pojo Class**

package MCA; import java.util.\*;

public class StrawHat { private String name;

private List<String> crewName; private Set<String> bounty;

private Map<String, String> ability;

public StrawHat(String name, 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 String toString() {

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/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>">

<!-- Bean for StrawHat with collection injection -->

<bean class="MCA.StrawHat" name="strawHat">

<constructor-arg name="name" value="The Straw Hat Pirates"/>

<!-- List of Crew Members -->

<constructor-arg name="crewName">

<list>

<value>Monkey D. Luffy</value>

<value>Roronoa Zoro</value>

<value>First son of the sea Jimbei</value>

<value>Vinsmoke Sanji</value>

<value>Demon child Nico Robin</value>

</list>

</constructor-arg>

<!-- Set of Bounties -->

<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>

<!-- Map of Abilities -->

<constructor-arg name="ability">

<map>

<entry key="luffy" value="rubber body"/>

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

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

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

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

</map>

</constructor-arg>

</bean>

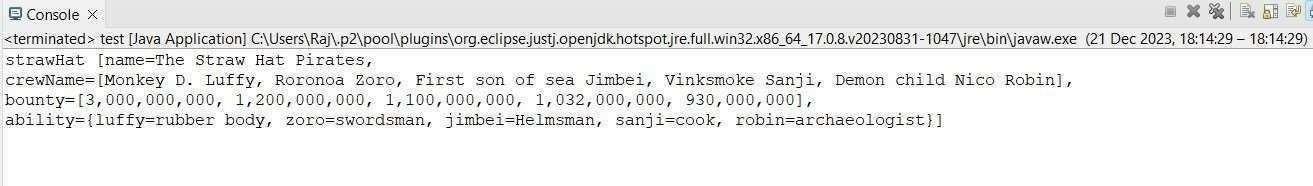
</beans>

**Main Class**

package MCA;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;



public class Test {

public static void main(String[] args) {

// Load the Spring context from the XML configuration ApplicationContext context = new

ClassPathXmlApplicationContext("MCA/applicationContext.xml");

StrawHat temp = (StrawHat) context.getBean("strawHat"); System.out.println(temp);

}

}

#### OUTPUT :

1. **Aim : Write a program to demonstrate Auto Wiring.**

**Pojo Class**

package MCA;

public class Chopper { private Zoro zoro;

// Getter for Zoro public Zoro getZoro() {

return zoro;

}

// Setter for Zoro

public void setZoro(Zoro zoro) { this.zoro = zoro;

}

public Chopper(MCA.Zoro zoro) { super();

this.zoro = zoro;

}

// Default constructor public Chopper() {

super();

}

// toString method @Override

public String toString() {

return "Chopper[Zoro=" + zoro + "]";

}

}

**Reference Class**

package MCA; public class Zoro {

private String name;

private double height; private int swords;

// Getter and Setter methods

public String getName() { 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;

}

public Zoro(String name, double height, int swords) { super();

this.name = name; this.height = height; this.swords = swords;

}

// Default constructor public Zoro() {

super();

}

// toString method @Override

public String toString() {

return "name of Character=" + name + ", height of Character=" + height + ", No. of swords=" + swords;

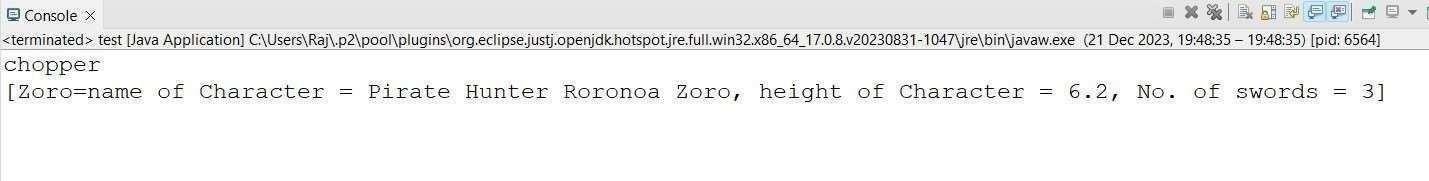
}

}

**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/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>">

<!-- Define Zoro Bean -->

<bean class="MCA.Zoro" name="Zoro" p:name="PirateHunter Roronoa Zoro" p:height="6.2" p:swords="3" />

<!-- Define Chopper Bean with autowiring byType -->

<bean class="MCA.Chopper" name="chopper" autowire="byType" />

</beans>

**Main Class**

package MCA;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Test {

public static void main(String[] args) { ApplicationContext context = new

ClassPathXmlApplicationContext("MCA/mcaConfig.xml");

Chopper temp = (Chopper) context.getBean("chopper"); System.out.println(temp);

}

}

**OUTPUT :**

# PRACTICAL NO. 8

1. **Aim: 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/4.0.0>

<http://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](http://maven.apache.org/)</url>

<properties>

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

</properties>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.2.3.RELEASE</version>

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.2.3.RELEASE</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.2.3.RELEASE</version>

</dependency>

<!-- AspectJ Runtime -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjrt</artifactId>

<version>1.9.7</version>

</dependency>

<!-- AspectJ Weaver -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.9.6</version>

</dependency>

<!-- JUnit -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**Interface (Guitar)**

package aop;

public interface Guitar { void makeSong();

}

**Target Object (Brook)**

package aop;

public class Brook implements Guitar { @Override

public void makeSong() { System.out.println("Song Started"); System.out.println("Song Ended");

}

}

**Aspect Class**

package aop;

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

public class McaAspect {

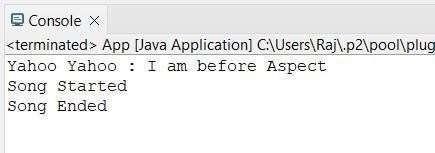
@Before("execution(\* aop.Brook.makeSong(..))") public void beforeSong() {

System.out.println("Yahoo Yahoo: I am before Aspect");

}

}

**Spring Configuration (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: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/>

<!-- Target Object Bean -->

<bean id="brook" class="aop.Brook"/>

<!-- Aspect Bean -->

<bean id="mcaAspect" class="aop.McaAspect"/>

</beans>

**Main Class()**

package aop;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext; public class App {

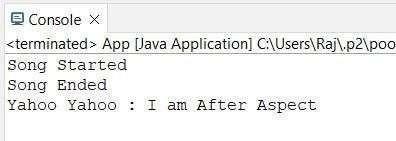
public static void main(String[] args) { ApplicationContext context = new

ClassPathXmlApplicationContext("aop/aopConfig.xml"); Guitar temp = (Guitar) context.getBean("brook"); temp.makeSong();

}

}

#### OUTPUT



1. **Aim: Write a program to demonstrate Spring AOP –after advice.**

**Aspect Class**

package aop;

import org.aspectj.lang.annotation.After;

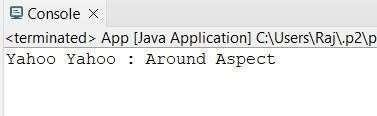
import org.aspectj.lang.annotation.AfterReturning; import org.aspectj.lang.annotation.AfterThrowing; import org.aspectj.lang.annotation.Around;

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

@Aspect publicclassmcaAspect{ @After("execution(\*brook.makeSong())")

public void afterSong() { System.out.println("Yahoo Yahoo: I am After Aspect"); } }

#### OUTPUT :



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

**Aspect Class**

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 {

@Around("execution(\* aop.Brook.makeSong(..))") public void aroundSong(ProceedingJoinPoint joinPoint)

{

try {

System.out.println("Yahoo Yahoo: Before method execution (Around Advice)");

// Proceed to the target method joinPoint.proceed();

System.out.println("Yahoo Yahoo: After method execution (Around Advice)");

} catch (Throwable e) {

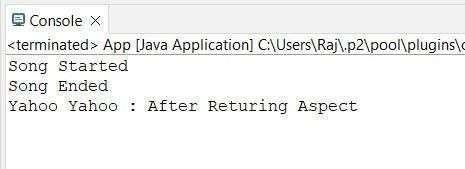
System.out.println("Yahoo Yahoo: Exception caught in Around Advice");

}

}

}

#### OUTPUT :



1. **Aim: -Write a program to demonstrate SpringAOP–after returning advice.**

**Aspect Class**

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(\* aop.Brook.makeSong(..))") public void afterReturnSong() {

System.out.println("Yahoo Yahoo: After Returning Aspect");

}

}

#### OUTPUT :

1. **Aim: Write a program to demonstrate Spring AOP –after throwing advice.**

**Aspect Class**

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(\* aop.Brook.makeSong(..))") private void selectAll() {}

@AfterThrowing(pointcut = "selectAll()", throwing = "error") public void afterThrowingAdvice(IllegalArgumentException error)

{

System.out.println("Yahoo Yahoo: There has been an exception: " + error.getMessage());

}

}

### Target Class

package aop;

public class Brook implements Guitar { @Override

public void makeSong() { System.out.println("Song Started"); System.out.println("Song Ended");

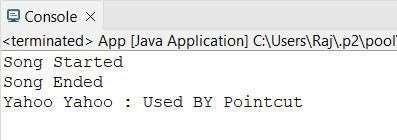
throw new IllegalArgumentException("An error occurred while making the song.");

}

}



#### OUTPUT :



**[F]Aim: Write a program to demonstrate Spring AOP – pointcuts**.

**AspectClass**

packageaop;

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 publicclassmcaAspect{

@Pointcut("execution(\*brook.makeSong())") publicvoidsongPointCut(){ System.out.println("YahooYahoo:Iampointcut");

}

@AfterReturning("songPointCut()") publicvoidafterSong(){ System.out.println("YahooYahoo:UsedBYPointcut");

}

}

#### OUTPUT:

**PRACTICAL NO. 9**

**[A] Aim: Write a program to insert, update and delete records from the given table.**

**Pom.xml**

<projectxmlns="<http://maven.apache.org/POM/4.0.0>"xmlns:xsi="<http://www.w3.org/2001/X> MLSchema-instance" xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0><http://maven.apache.org/xsd/mave> n- 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](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**

<?xmlversion="1.0"encoding="UTF-8"?>

<beans xmlns="<http://www.springframework.org/schema/beans>"xmlns:xsi="<http://www.w3.org/200> 1/XMLSchema-instance"

xmlns:context="<http://www.springframework.org/schema/context>"xmlns:p="http://www.spri ngframework.org/schema/p"xmlns:c="<http://www.springframework.org/schema/c>"xsi:schem aLocation="<http://www.springframework.org/schema/beans>[http://www.springframework.org](http://www.springframework.org/)

/schema/beans/spring- beans.xsd<http://www.springframework.org/schema/contex>[thttp://www.springframework.org/](http://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:dataSour ce- ref="ds"/>

</beans>

**Main Class**

packagecom.mca; importorg.springframework.context.ApplicationContext;

importorg.springframework.context.support.ClassPathXmlApplicationContext; importorg.springframework.jdbc.core.JdbcTemplate;

publicclassApp

{

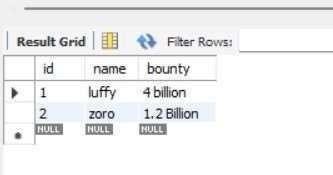
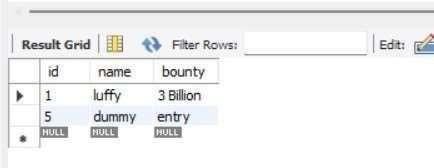
publicstaticvoidmain(String[]args)

{

System.out.println("kaizokuoniorewanaru!");

ApplicationContext context= new ClassPathXmlApplicationContext("com/mca/config.xml"); JdbcTemplate temp =context.getBean("jdbcTemplate",JdbcTemplate.class);

// insertQuery Stringquery1="insertintostrawHatvalues(?,?,?)";Stringquery2="updatestrawHatsetbounty=?w



hereid=?"; String query3 = "delete from strawHatwhereid=?";

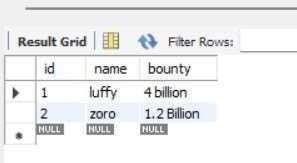
// firequery intresult1=temp.update(query1,2,"zoro","1.2Billion");System.out.println("Numberofrecords insetred "+ result1); intresult2=temp.update(query2,"4billion",1);System.out.println("Numberofrecordsupdated "+ result2);

intresult3=temp.update(query3,5);System.out.println("NumberofrecordsDeleted "+ result3);

}

}

#### OUTPUT:



**[B] Aim: Write a program to demonstrate Prepared Statement in Spring JDBC Template.**

**MainClass**

packagecom.mca;

importjava.sql.Connection;import java.sql.PreparedStatement; import java.sql.SQLException; importorg.springframework.context.ApplicationContext; importorg.springframework.context.support.ClassPathXmlApplicationContext; importorg.springframework.jdbc.core.JdbcTemplate; importorg.springframework.jdbc.core.PreparedStatementCreator; publicclassApp

{

publicstaticvoidmain(String[]args)

{

System.out.println("kaizokuoniorewanaru!");

ApplicationContext context= new ClassPathXmlApplicationContext("com/mca/config.xml"); JdbcTemplate temp = context.getBean("jdbcTemplate", JdbcTemplate.class); Stringquery1="insertintostrawHat(id,name,bounty)values(?,?,?)"; intresult=temp.update(newPreparedStatementCreator(){

@Override publicPreparedStatementcreatePreparedStatement(Connectioncon)throws SQLException{

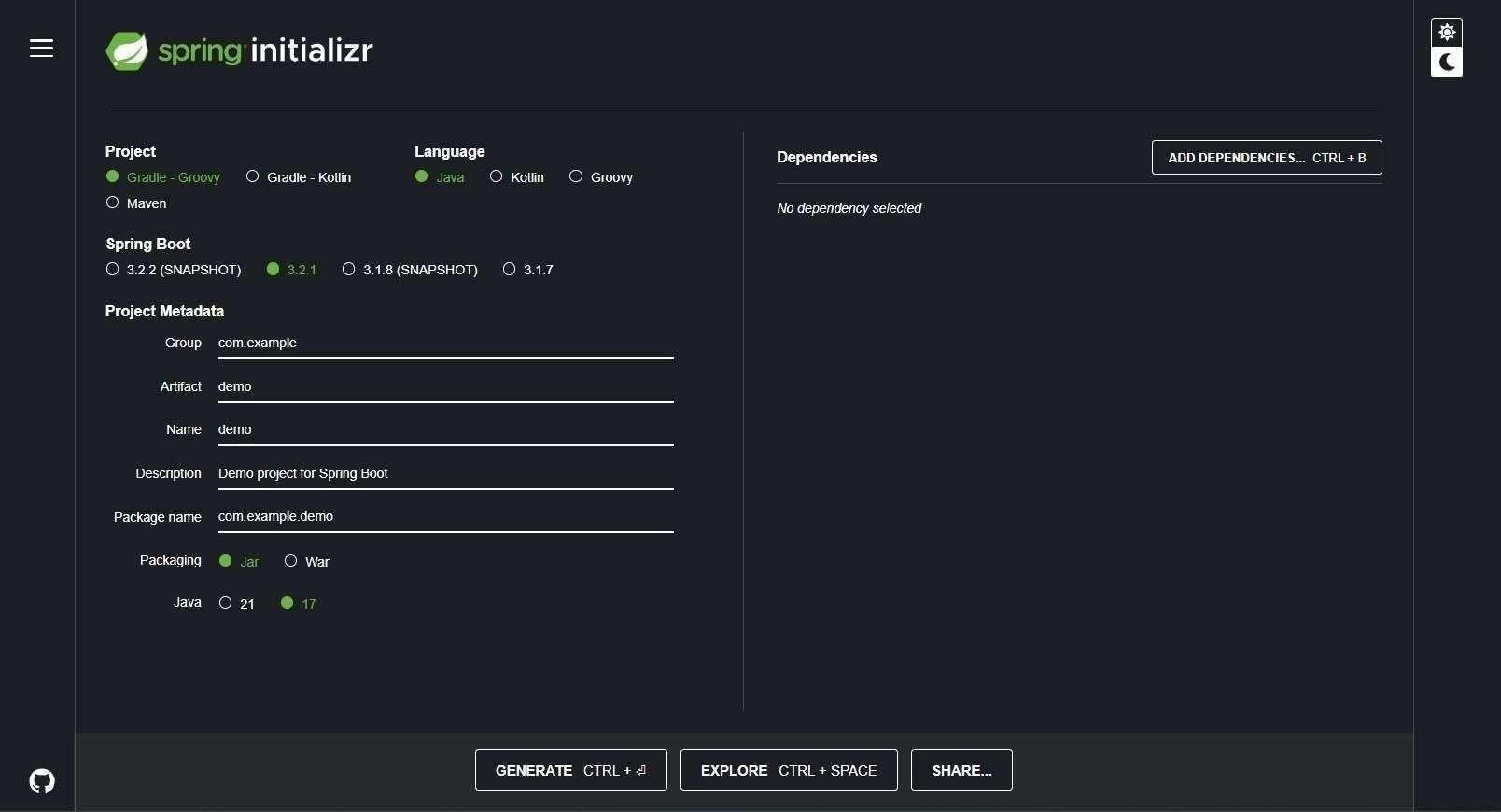
PreparedStatementps=con.prepareStatement(query1); ps.setInt(1, 3);

ps.setString(2, "zoro"); ps.setString(3,"1.1Billion"); returnps;

}

});

#### OUTPUT :



**PRACTICAL NO. 10**

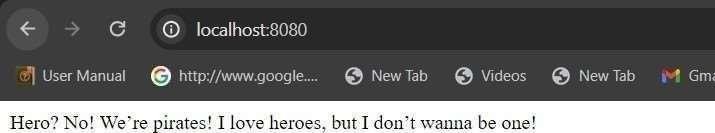
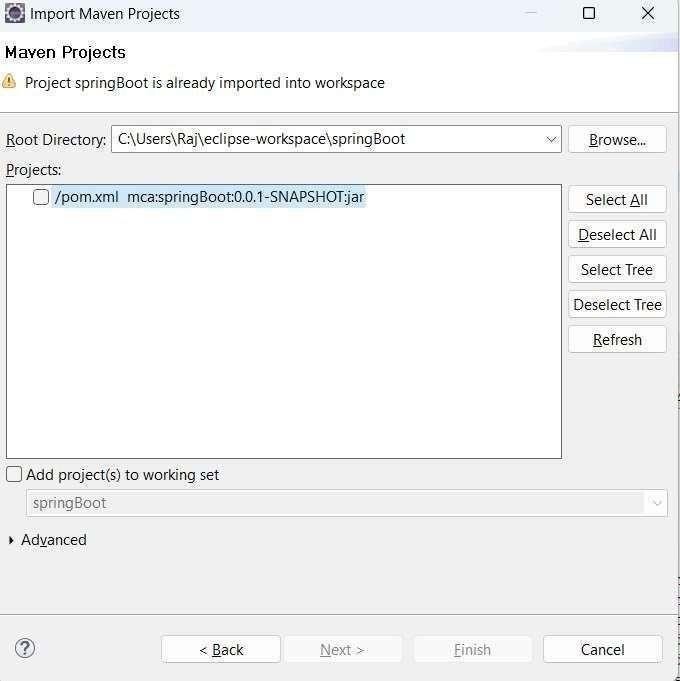
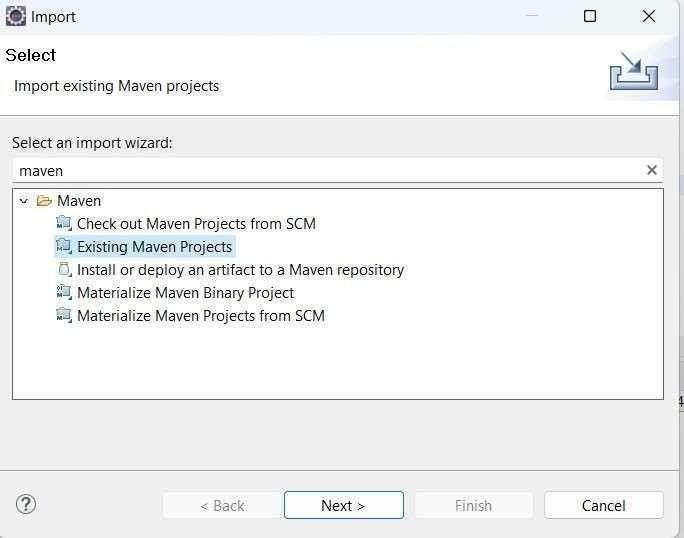
**[A] Aim: Write a program to create a simple Spring Boot application that print message.**

**Step1:**

Go to Spring Initializr. Select the type of project (Maven). Choosethelanguage(Java).SelecttheSpringBootversion.

Fillin theproject metadata. Add thenecessary dependencies (atleast spring- boot- starter-web). Click on “Generate” to download the project.

**Step2:** OpenEclipseIDE.NavigatetoFile>Import.Select“ExistingMavenProjects”.Clickon“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”.



### MainClass:

packagecom.mca.spring;

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

@SpringBootApplication publicclassmyApplication{

publicstaticvoidmain(String[]args){SpringApplication.run(myApplication.class,args);

}

beone!";

}

@RestController

public class controller { @GetMapping("/") publicStringquote(){ return"Hero?No!We’repirates!Iloveheroes,butIdon’twanna

}

}

**OUTPUT :**