**Practical No: 1**

**Aim: Assignments on Java Generics.**

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

**Code:**

**package** generic;

**public** **class** main {

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

GenericsClass<Integer> intObj = **new** GenericsClass<>(5);

System.***out***.println("Generic Class returns: " + intObj.getData());

GenericsClass<String> stringObj = **new** GenericsClass<>("Java Programming");

System.***out***.println("Generic Class returns: " + stringObj.getData());

}

}

**class** GenericsClass<T> {

**private** T data;

**public** GenericsClass(T data) {

**this**.data = data;

}

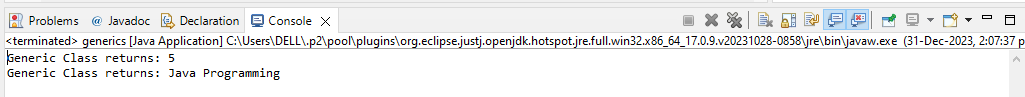
**public** T getData() {

**return** **this**.data;

}

}

**Output:**



**2. Write a Java Program to demonstrate Generic Methods.**

**Code:**

**package** generic;

**public** **class** genericmethods {

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

DemoClass demo = **new** DemoClass();

demo.<String>genericsMethod("Java programming to demonstrate Generic Methods");

demo.<Integer>genericsMethod(50);

}

}

**class** DemoClass {

**public** <T> **void** genericsMethod(T data) {

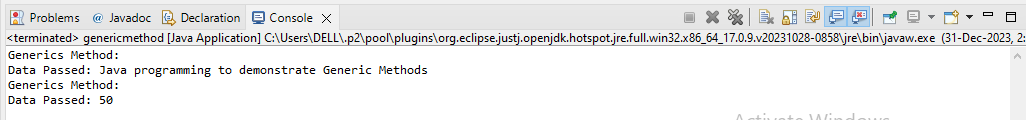
System.***out***.println("Generics Method:");

System.***out***.println("Data Passed: " + data);

}

}

**Output:**



**3. Write a Java Program to demonstrate Wildcards in Java Generics.**

**1) UpperBounded:**

package MCA;

import java.util.\*;

public class Prac1C {

public static void printList(List<?> list) {

for (Object elem : list)

System.out.print(elem + " ");

System.out.println();

}

// method that sums the elements of a list of numbers

public static double sumOfList(List<? extends Number> list) {

double s = 0.0;

for (Number n : list)

s += n.doubleValue();

return s;

}

public static void main(String[] args) {

List<Integer> integerList = Arrays.asList(1, 2, 3, 4, 5);

System.out.println("integerList contains: ");

printList(integerList);

System.out.println("The sum is: " + sumOfList(integerList));

List<Double> doubleList = Arrays.asList(1.1, 2.2, 3.3, 4.4, 5.5);

System.out.println("\ndoubleList contains: ");

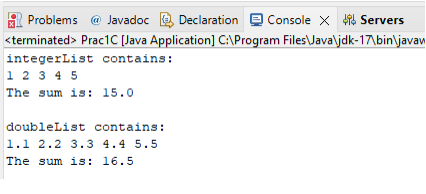
printList(doubleList);

System.out.println("The sum is: " + sumOfList(doubleList));

}

}

**Output:**



**2) Lowerbound**

package WildCard;

import java.util.Arrays;

import java.util.List;

public class lowerbound {

public static void main(String[] args) {

List<Integer>list1=Arrays.asList(5,8,6,7,11);

superClass(list1);

List<Number>list2=Arrays.asList(4,8,55,7,41);

superClass(list2);

}

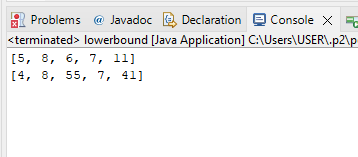
public static void superClass(List<? super Integer>list) {

System.out.println(list);

}

}

**Output:**



**3) UpperBounded**

package WildCard;

import java.util.\*;

public class upperbound<T> {

private static Number sum(List<? extends Number>numbers) {

double a=0.0;

for(Number n:numbers)

a+=n.doubleValue();

return a;}

public static void main(String[] args) {

List<Integer>list1=Arrays.asList(5,8,6,7,11);

System.out.println(sum(list1));

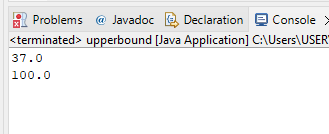
List<Double>list2=Arrays.asList(4.2,8.5,5.5,77.7,4.1);

System.out.println(sum(list2));

}

}

**Output:**



**Practical No: 2**

**Aim: Assignments on List Interface.**

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

**Code:**

package collection;

import java.util.\*;

public class ArrayListDemo {

public static void main(String args[]) {

ArrayList<String> list = new ArrayList<String>();

list.add("hello");

list.add("dear");

list.add("root");

System.out.println("ArrayList:" + list);

//System.out.println("root is at index: " + list.indexOf(10));

for(String s:list){

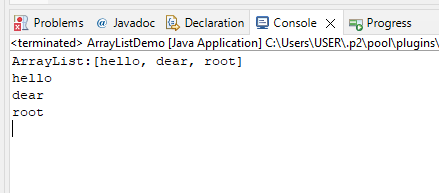
System.out.println(s);

}

}

}

**Output:**



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

**Code:**

package collection;

import java.util.ArrayList;

import java.util.Iterator;

public class ReverseArrayList {

public static void main(String[] args) {

ArrayList<String> arr = new ArrayList<String>();

arr.add("hello");

arr.add("dear");

arr.add("root");

System.out.println(arr);

Iterator<String> it= arr.iterator();

for (int i = arr.size()-1; i >=0; i--) {

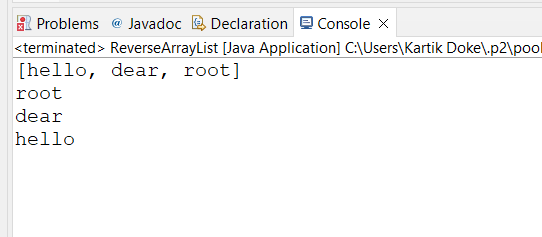
System.out.println(arr.get(i));

}

}

}

**Output:**

****

**Practical No: 3**

**Aim:** **Assignments on Set Interface.**

**1. 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**

**Code:**

**package** collectionEx;

**import** java.util.\*;

**public** **class** disha\_p {

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

Set<String> items = **new** HashSet<>(Arrays.*asList*("Chirantan", "Disha",

"Tanaji", "Rupali", "Mane"));

// Get an Iterator for the set

Iterator<String> iterator = items.iterator();

System.***out***.println("Printing items in the set:");

**while** (iterator.hasNext()) {

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

}

// Convert the Set to a List

List<String> itemList = **new** ArrayList<>(items);

// Reverse the list

Collections.*reverse*(itemList);

System.***out***.println("\nPrinting items in reverse order:");

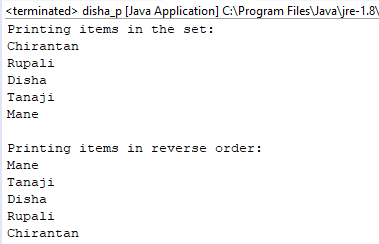
**for** (String item : itemList) {

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

}}

}

**Output:**

****

**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**

**Code:-**

**package** collectionEx;

**import** java.util.\*;

**public** **class** disha\_p {

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

Set<String> set1 = **new** HashSet<>();

set1.add("DISHA");

set1.add("TANAJI");

set1.add("MANE");

System.***out***.println("Set1: " + set1);

// b. Insert items of one set into another set

Set<String> set2 = **new** HashSet<>();

set2.add("CHIRANATAN");

set2.add("SAWANT");

set1.addAll(set2);

System.***out***.println("After adding set2 into set1, set1: " + set1);

// c. Remove items from the set

set1.remove("SHREESHA");

System.***out***.println("After removing 'Date' from set1, set1: " + set1);

// d. Search the specified item in the set

String item = "VEDIKA";

**if** (set1.contains(item)) {

System.***out***.println("'" + item + "' is found in a set.");

} **else** {

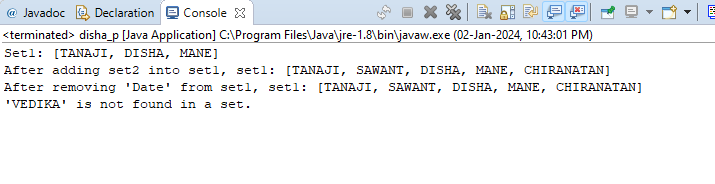
System.***out***.println("'" + item + "' is not found in a set.");

}

}

}

OUTPUT:-



**Practical No:4**

**Aim: Assignments on Map Interface**

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

**a. Add items in the map.**

**b. Remove items from the map**

**c. Search specific key from the map**

**d. Get value of the specified key**

**e. Insert map elements of one map in to other map.**

**f. Print all keys and values of the map**

**Code:**

**package** collectionEx;

**import** java.util.\*;

**public** **class** disha\_p {

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

// a. Add items in the map

Map<String, String> map1 = **new** HashMap<>();

map1.put("DISHA", "MANE");

map1.put("CHIRANTAN", "SONAVANE");

map1.put("SHREESHA", "MULIK");

System.***out***.println("A]\tMap1: " + map1);

// b. Remove items from the map

map1.remove("DISHA");

System.***out***.println("B]\tAfter removing 'Carrot' from map1, map1: " +

map1);

// c. Search specific key from the map

String key = "CHIRANATAN";

**if** (map1.containsKey(key)) {

System.***out***.println("C]\t'" + key + "' is found in the map.");

} **else** {

System.***out***.println("C]\t'" + key + "' is not found in the map.");

}

// d. Get value of the specified key

String value = map1.get(key);

System.***out***.println("D]\tThe value of '" + key + "' is: " + value);

// e. Insert map elements of one map into another map

Map<String, String> map2 = **new** HashMap<>();

map2.put("VEDIKA", "SAWANT");

map2.put("SONU", "LOKE");

map1.putAll(map2);

System.***out***.println("E]\tAfter adding map2 into map1, map1: " + map1);

// f. Print all keys and values of the map

**for** (Map.Entry<String, String> entry : map1.entrySet()) {

System.***out***.println("F]\tKey: " + entry.getKey() + ", Value: " +

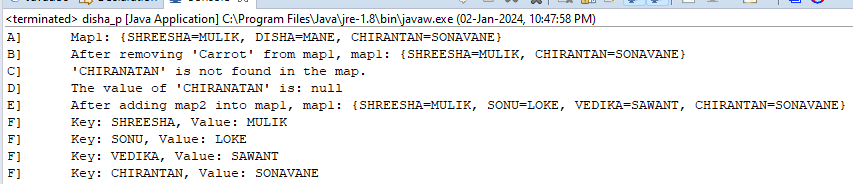
entry.getValue());

}

}

}

**Output:**

****

**Practical No: 5**

**Aim:** **Assignments on Lambda Expression.**

**1. Write a Java program using Lambda Expression to print ”Hello World”.**

**Code:**

**package** LambdaExpression;

**public** **class** P1 {

**interface** Hello {

**void** hello();

}

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

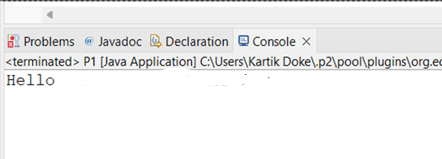
Hello obj = () -> System.***out***.println("Hello");

obj.hello();

}

}

**Output:**

****

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

**Code:**

**package** LambdaExpression;

//Q2. Write a java program to execute Lambda Expression with single parameter.

**interface** Sayable {

**public** String say(String name);

}

**public** **class** P2 {

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

Sayable S1 = (name) -> {

**return** "Hello," + name;

};

System.***out***.println(S1.say(" "));

Sayable S2 = (name) -> {

**return** " Disha your " + name;

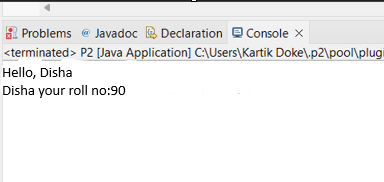
};

System.***out***.println(S2.say("roll no:90"));

}

}

**Output:**

****

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

**Code:**

**package** LambdaExpression;

**interface** Addition{

**int** add(**int** a, **int** b);

}

**public** **class** P3 {

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

Addition ad1=(a,b)->(a+b);

System.***out***.println("Addition is: "+ad1.add(50, 11));

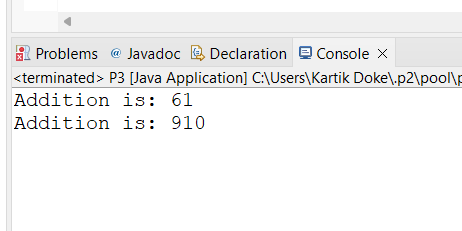
Addition ad2=(**int** x,**int** y)->(x+y);

System.***out***.println("Addition is: "+ad2.add(510,400));

}

}

**Output:**

****

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

**a. Convert Fahrenheit to Celcius**

package LambdaExpression;

import java.util.Scanner;

class FahrenheittoCelsius

{

double celsius(double f)

{

return (f-32)\*5/9;

}

public static void main (String arg[])

{

double a,c;

Scanner sc=new Scanner (System.in);

System.out.print("Enter Fahrenheit temperature: ");

a=sc.nextDouble();

FahrenheittoCelsius fah=new FahrenheittoCelsius ( ) ;

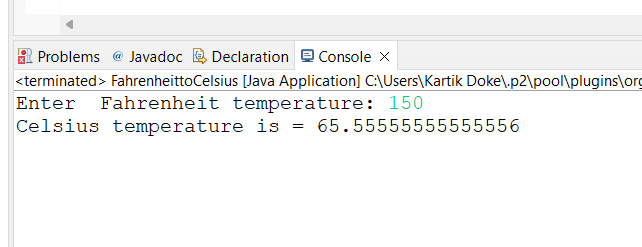
double result=fah.celsius(a);

System.out.println("Celsius temperature is = "+result);

}

}

**Output:**

****

**b. Convert Kilometers to Miles.**

package LambdaExpression;

import java.util.Scanner;

public class KilometertoMiles {

public static void main(String[] args) {

double kilometers, miles;

double conversionFactor = 1.609344;

System.out.print("Enter distance value in kilometer:");

Scanner input = new Scanner(System.in);

kilometers = input.nextDouble();

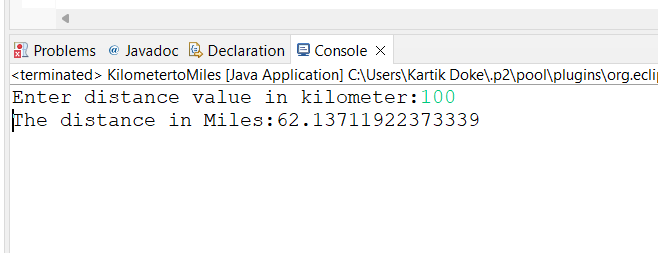
miles = kilometers / conversionFactor;

System.out.println("The distance in Miles:" + miles);

}

}

**Output:**

****

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

**Code:**

**package MCA;**

**@FunctionalInterface**

**interface Adder2 {**

**int add(int a, int b);**

**}**

**@FunctionalInterface**

**interface Subtractor {**

**int subtract(int a, int b);**

**}**

**public class Prac5E {**

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

**Adder2 adder = (a, b) -> a + b;**

**Subtractor subtractor = (a, b) -> a - b;**

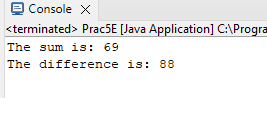
**int sum = adder.add(60, 9);**

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

**int difference = subtractor.subtract(90, 2);**

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

**Output:**

****

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

**Code:**

**package** collectionEx;

**import** java.util.\*;

@FunctionalInterface

**interface** ConcatenateInterface {

String concatenate(String a, String b);

}

**public** **class** disha\_p{

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

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

String str1 = "Disha ";

String str2 = "Mane";

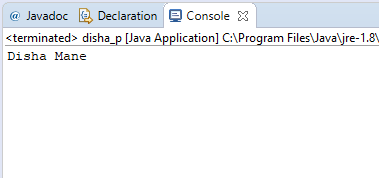
String result = concat.concatenate(str1, str2);

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

}

}

**Output:**

****

**Practical No: 6**

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

**1. Write Programs to demonstrate different Implicit Objects**

**a. OUT**

**b. Request**

**c. Session**

**Code:**

<%@ 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 ordeal");%>

<h1>Reuqest 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");

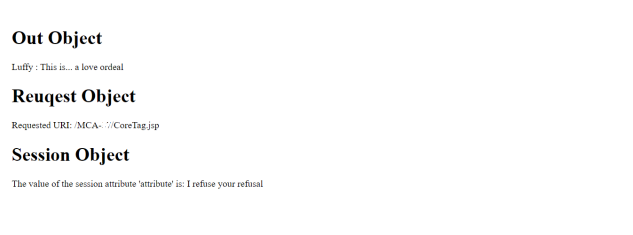
out.println("The value of the session attribute 'attribute' is: " +attribute);

%>

</body>

</html>

**Output:**

****

**2. Write Programs to demonstrate temporary storage using Bean.**

**Code:**

<%@ 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>

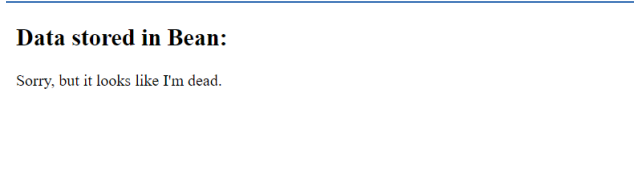
<h2>Data stored in Bean:</h2>

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

</body>

</html>

**Output:**

****

**3. Write a program to demonstrate Standard Action tags**

**Code:**

<%@ 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>

<body>

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

<%-- JSP Declaration --%>

<%! **int** count = 0; %>

<%-- JSP Scriptlet --%>

<%

count++;

out.println("This is a 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>

</body>

</html>

**Output:**

## **4.** Write a program to demonstrate JSP Directives

**Code:**

<%@ 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"*](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>

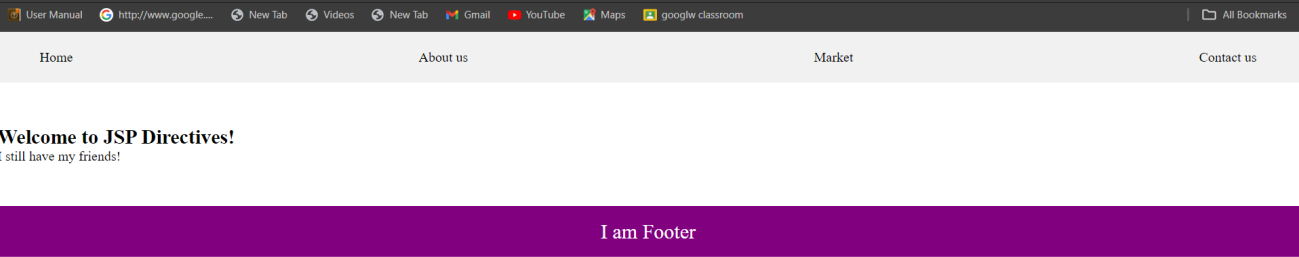
### <c:out value=*"*${'I still have my friends!'}*"*/>

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

### </body>

</html>

# Output :



## **5.Write a program to demonstrate Session Tracking using Cookies**

**Code:**

<%@ 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 :

**6. Write a program to demonstrate JSTL Tags**

<%@ taglib uri=[*"http://java.sun.com/jsp/jstl/core"*](http://java.sun.com/jsp/jstl/core) prefix=*"c"* %>

<%@ taglib uri=[*"http://java.sun.com/jsp/jstl/fmt"*](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 :

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

**Code:**

<%@ page import = "java.io.\*,java.util.\*,java.sql.\*"%>

<%@ page import = "javax.servlet.http.\*,javax.servlet.\*" %>

<%@ taglib uri="http://java.sun.com/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>

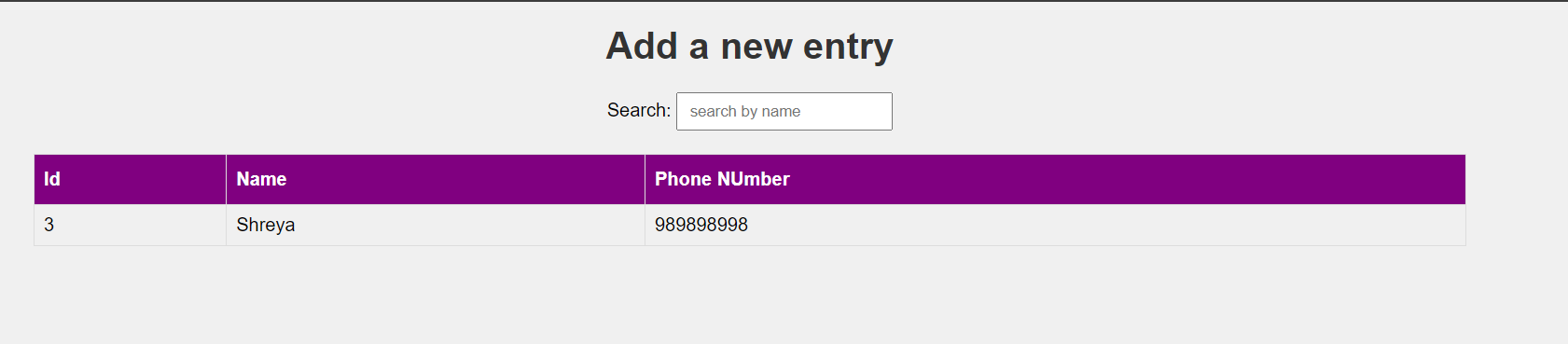
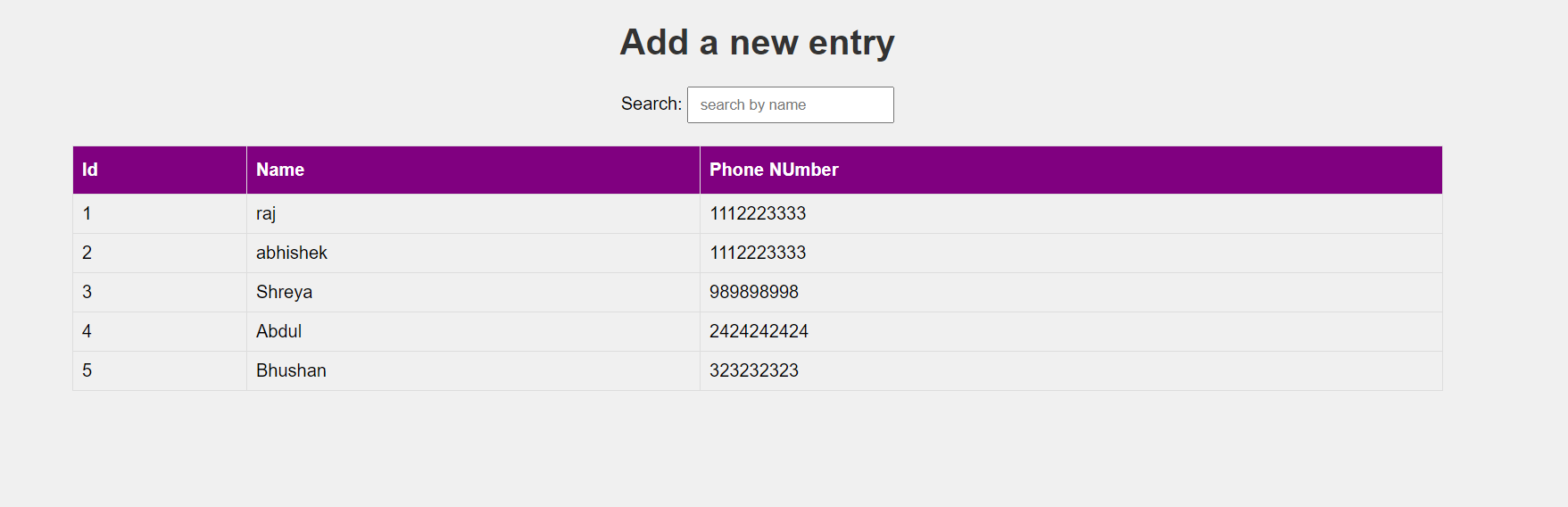
</c:forEach>

</table>

</body>

</body>

</html>

**Output:**

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

**Code:**

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

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html>

<html>

<head>

<meta 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">

<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>

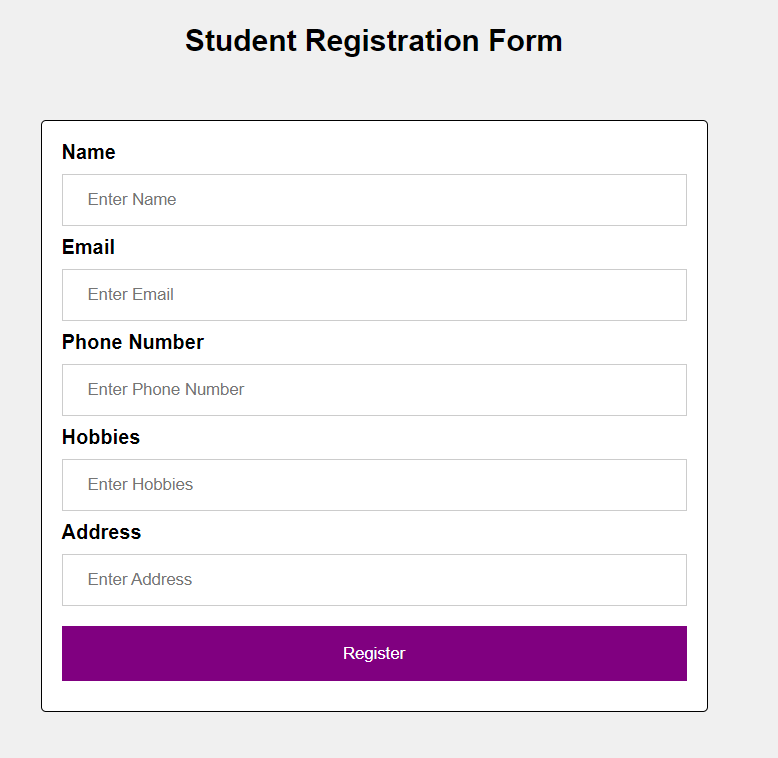
<button type="submit">Register</button>

</div>

</body>

</html>

**Output:-**



**PRACTICAL 7**

**AIM:ASSIGNMENT BASED SPRING FRAMEWORK**

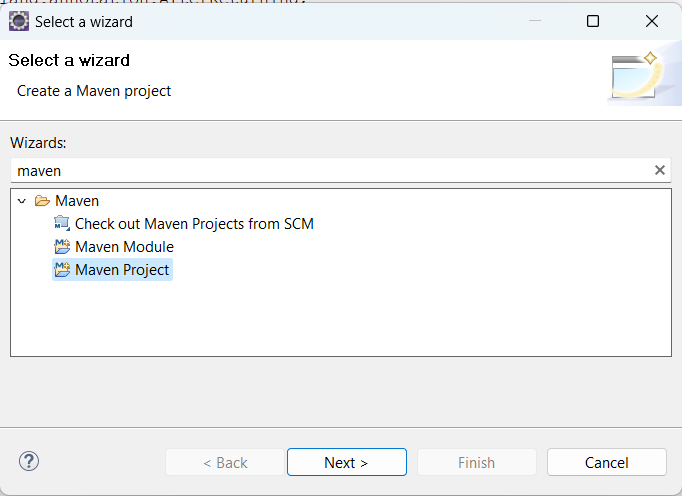
**A.Write a program to print Singer Name and Age using spring framework.**

**CODE:**

**Maven project**

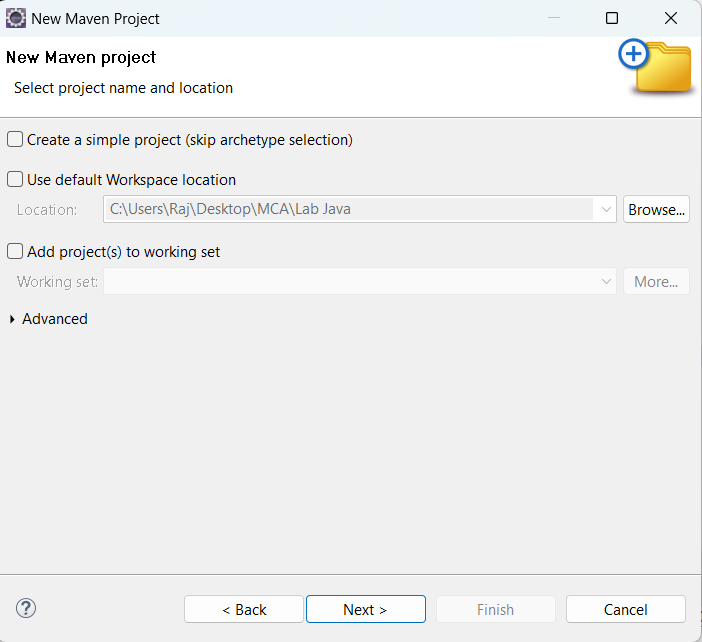
**1) Open Eclipse IDE, Navigate to File, then New, then Others., Select Maven Project,**

**Click on the “Next” button.**



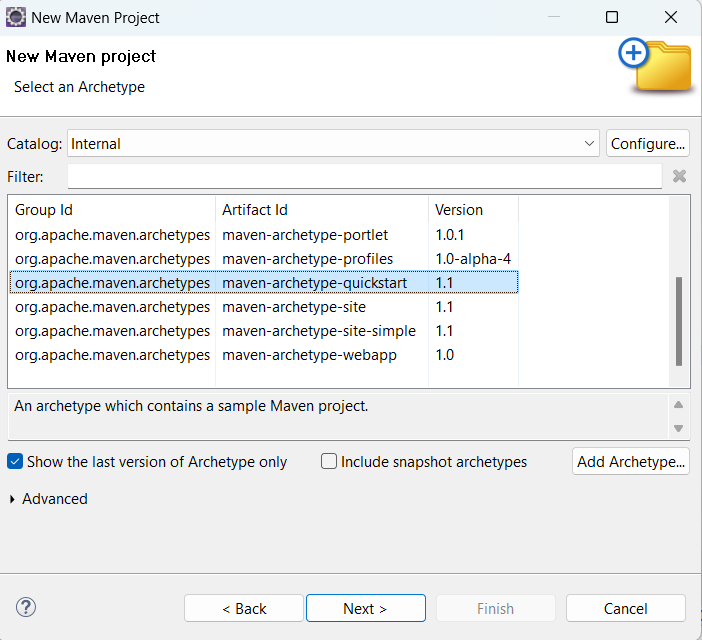
**2) Check the option ‘use default workspace location or choose your desired workspace**

**location.**



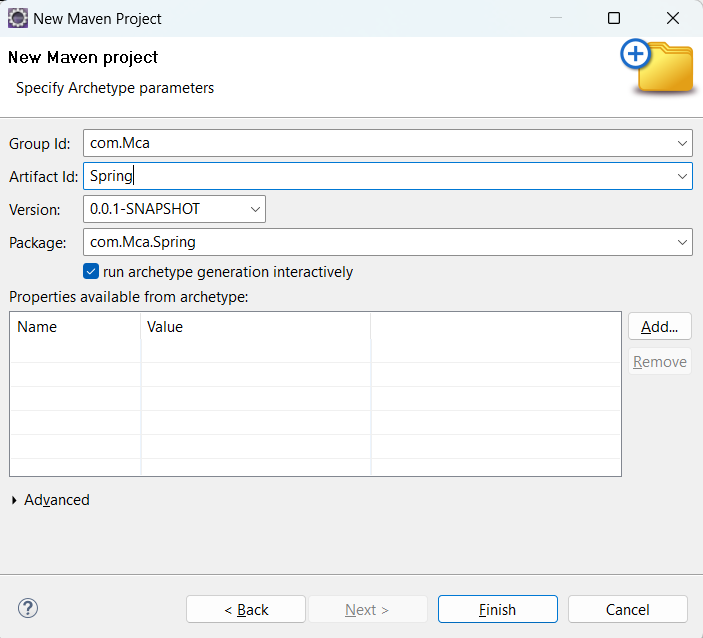
**3) Select catalog Internal the archetype ‘maven-archetype-quickstart’.Click on the**

**“Next” button**



**4) Enter your project’s Group Id. Enter your project’s Artifact Id. Click on the**

**“Finish” button.**



**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</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>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

</project>

**POJO class**

**package** MCA;

**public class** Singer {

**private** String name;

**private** Integer age;

**public** String getName() {

**return** name;

}

**public void** setName(String name) {

**this**.name = name;

}

**public** Integer getAge() {

**return** age;

}

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

**this**.age = age;

}

**public** Singer(String name, Integer age) {

**super**();

**this**.name = name;

**this**.age = age;

}

**public** Singer() {

**super**();

// **TODO** Auto-generated constructor stub

}

@Override

**public** String toString() {

**return** "Singer [name=" + name + ", age=" + age + "]";

}

}

**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: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 class=*"MCA.Singer"* name=*"singer"* p:Name=*"Luffy"* p:Age=*"19"*/>

</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");

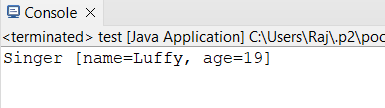
Singer temp = (Singer) context.getBean("singer");

System.out.println(temp);

}

}

**Output :**



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

**POJO Class**

**package** MCA;

**public class** Zoro {

**private** String name;

**private double** height;

**private int** swords;

//

setter and getter 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

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

**super**();

**this**.name = name;

**this**.height = height;

**this**.swords = swords;

}

**public** Zoro() {

**super**();

}

//

to string method

@Override

**public** String toString() {

**return** "name of Character = " + name + ", height of Character = "

+ height + ", No. of swords = " + swords ;

}

}

**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: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 class=*"MCA.Zoro"* name=*"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");

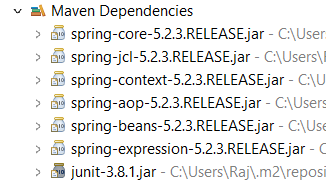
Zoro temp = (Zoro) context.getBean("zoro");

System.out.println(temp);

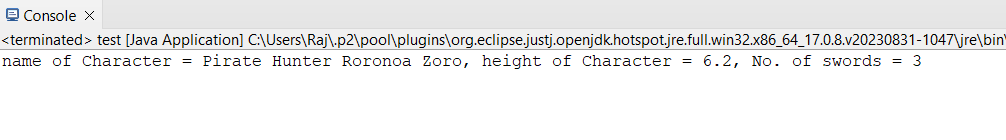
}

}

**Maven Dependencies**



**Output :**



**C.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** " Charactername = " + name + ", No. of gears = " + gears +

", height = " + height + "]";

}

}

**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: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 class=*"MCA.luffy"* name=*"luffy"* c:name=*"Monkey D. Luffy"* c:height=*"5.8"*

c:gears=*"5"*/>

</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");

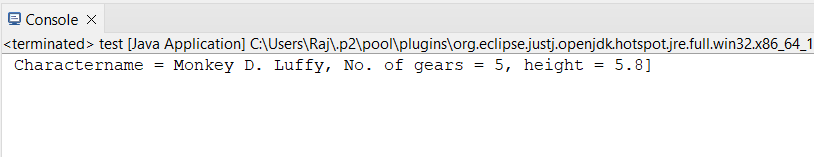
luffy temp = (luffy) context.getBean("luffy");

System.out.println(temp);

}

}

**Output :**



**D.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;

**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;

}

**public** sanji(String name, **double** height, Zoro obj) {

**super**();

**this**.name = name;

**this**.height = height;

**this**.obj = obj;

}

**public** sanji() {

**super**();

// **TODO** Auto-generated constructor stub

}

@Override

**public** String toString() {

**return** "sanji [name=" + name + ", height=" + height + ", \nobj="

+ obj + "]";

}

}

**Reference class**

**package** MCA;

**public class** Zoro {

**private** String name;

**private double** height;

**private int** swords;

//

setter and getter 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

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

**super**();

**this**.name = name;

**this**.height = height;

**this**.swords = swords;

}

**public** Zoro() {

**super**();

}

//

to string method

@Override

**public** String toString() {

**return** "name of Character = " + name + ", height of Character = "

+ height + ", No. of swords = " + swords ;

}

}

**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: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 class=*"MCA.Zoro"* name=*"zoro"* p:name=*"Pirate Hunter Roronoa Zoro"*

p:height=*"6.2"* p:swords=*"3"*/>

<bean class=*"MCA.sanji"* name=*"sanji"* p:name=*"Vinsmoke Sanji"* p:height=*"6.0"*

p:obj-ref=*"zoro"*/>

</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");

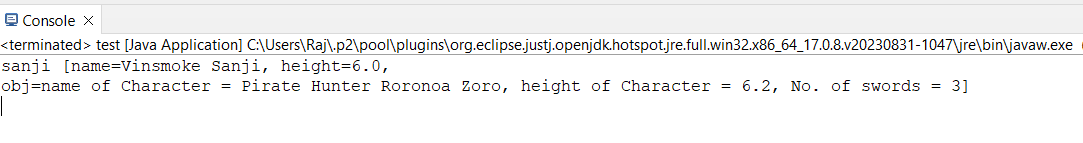
sanji temp = (sanji) context.getBean("sanji");

System.out.println(temp);

}

}

**Output :**



**E. 5.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;

@Override

**public** String toString() {

**return** "ussop [Name=" + Name + ", height=" + height + ", \nobj="

+ obj + "]";

}

**public** ussop(String name, **double** height, luffy obj) {

**super**();

Name = name;

**this**.height = height;

**this**.obj = obj;

}

}

**Reference 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** " Charactername = " + name + ", No. of gears = " + gears +

", height = " + height + "]";

}

}

**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: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 class=*"MCA.luffy"* name=*"luffy"* c:name=*"Monkey D. Luffy"* c:height=*"5.8"*

c:gears=*"5"*/>

<bean class=*"MCA.ussop"* name=*"ussop"* c:name=*"Sogeking Ussop"* c:height=*"5.11"*

c:obj-ref=*"luffy"*/>

</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");

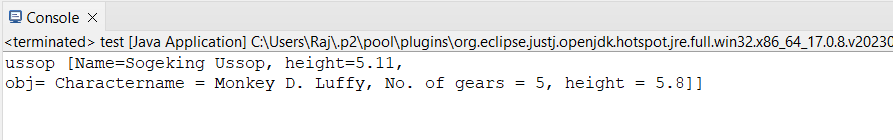
ussop temp = (ussop) context.getBean("ussop");

System.out.println(temp);

}

}

**Output :**



**F. 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 + "]";

}

}

**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: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 class=*"MCA.strawHat"* name=*"strawHat"*>

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

<constructor-arg name=*"crewName"*><list>

<value>Monkey D. Luffy</value>

<value>Roronoa Zoro</value>

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

<value>Vinksmoke Sanji</value>

<value>Demon child Nico Robin</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=*"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) {

ApplicationContext context = **new**

ClassPathXmlApplicationContext("MCA/mcaConfig.xml");

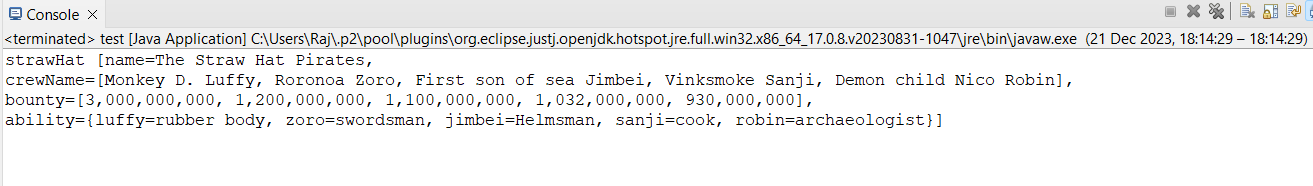
strawHat temp = (strawHat) context.getBean("strawHat");

System.out.println(temp);

}

}

**Output :**



**G.Write a program to demonstrate Autowiring**

**POJO class**

**package** MCA;

**public class** chopper {

**private** Zoro Zoro;

**public** Zoro getZoro() {

**return** Zoro;

}

**public void** setZoro(Zoro zoro) {

Zoro = zoro;

}

**public** chopper(MCA.Zoro zoro) {

**super**();

Zoro = zoro;

}

**public** chopper() {

**super**();

}

@Override

**public** String toString() {

**return** "chopper [Zoro=" + Zoro + "]";

}

}

**Reference Class**

**package** MCA;

**public class** Zoro {

**private** String name;

**private double** height;

**private int** swords;

//

setter and getter 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

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

**super**();

**this**.name = name;

**this**.height = height;

**this**.swords = swords;

}

**public** Zoro() {

**super**();

}

//

to string method

@Override

**public** String toString() {

**return** "name of Character = " + name + ", height of Character = "

+ height + ", No. of swords = " + swords ;

}

}

**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: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 class=*"MCA.Zoro"* name=*"Zoro"* p:name=*"Pirate Hunter Roronoa Zoro"*

p:height=*"6.2"* p:swords=*"3"*/>

<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");

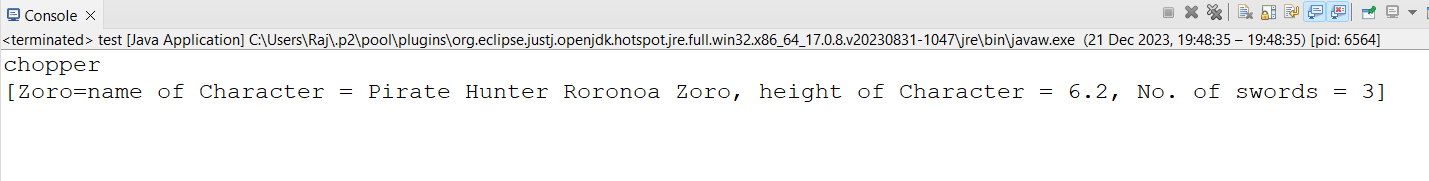
System.out.println(temp);

}

}

**Output :**

x



**PRACTICAL 8**

**ASSIGNMENT BASED ASPECT ORIENTED PROGRAMMING**

**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/4.0.0 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</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();

}

**Target Object**

**package** aop;

**public class** brook **implements** Guitar {

**public void** makeSong() {

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

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

}

}

**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 {

@Before("execution(\* brook.makeSong())")

**public void** beforeSong() {

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

}

}

**Configuration class**

<?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 "*>

<aop:aspectj-autoproxy/>

<bean name=*"brook"* class=*"aop.brook"*/>

<bean name=*"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");

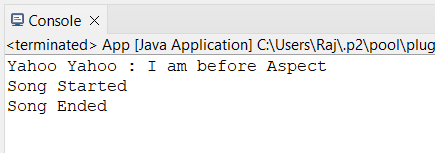
Guitar temp = (Guitar) context.getBean("brook");

temp.makeSong();

}

}

**Output :**



**B. 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

**public class** mcaAspect {

@After("execution(\* brook.makeSong())")

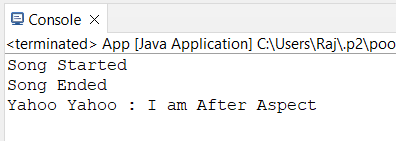
**public void** afterSong() {

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

}

}

**Output :**



**C.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(\* brook.makeSong())")

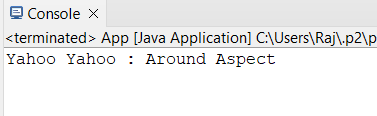
**public void** aroundSong() {

System.***out***.println("Yahoo Yahoo : Around Aspect");

}

}

**Output :**



**D.Write a program to demonstrate Spring AOP – 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

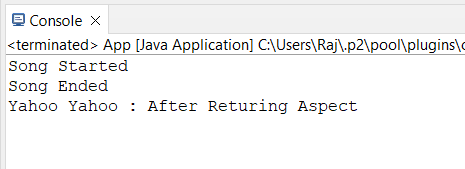
**public class** mcaAspect {

@AfterReturning("execution(\* brook.makeSong())")

**public void** AfterReturnSong() {

System.***out***.println("Yahoo Yahoo : After Returing Aspect"); } }

**Output :**



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

**Target Class :**

**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 occurred while making

the song.");

}

}

**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(\* brook.makeSong(..))")

**private void** selectAll(){}

@AfterThrowing(pointcut = "selectAll()", throwing= "error")

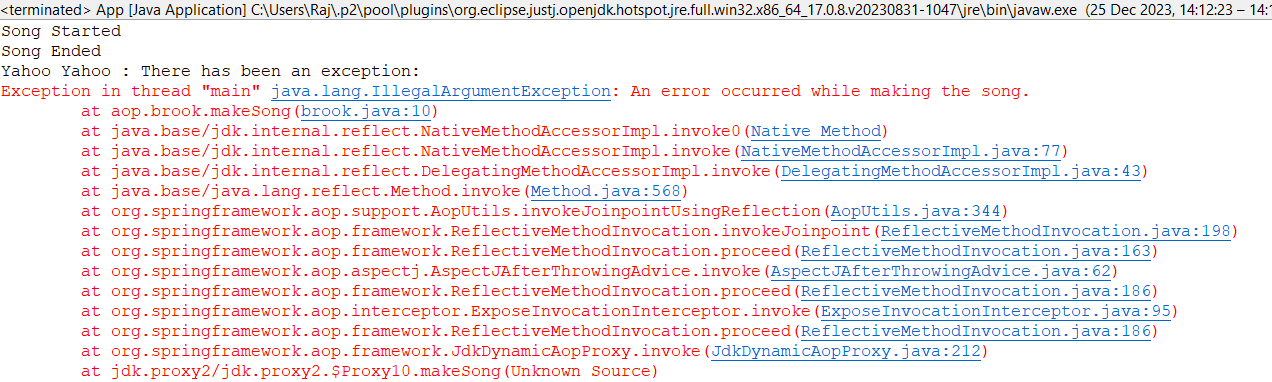
**public void** afterThrowingAdvice(IllegalArgumentException error){

System.***out***.println("Yahoo Yahoo : There has been an exception: ");

}

}

**Output :**



**F. Write a program to demonstrate Spring AOP – pointcuts.**

**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(\* brook.makeSong())")

**public void** songPointCut() {

System.***out***.println("Yahoo Yahoo : I am pointcut ");

}

@AfterReturning("songPointCut()")

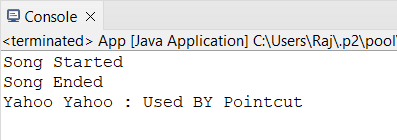
**public void** afterSong() {

System.***out***.println("Yahoo Yahoo : Used BY Pointcut");

}

}

**Output :**



**PRACTICAL 9**

**ASSIGNMENT BASED SPRING JDBC**

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

**Create table in mysql**

**create table student\_detail(id int not null,name varchar(50),address varchar(100));**

**insert into student\_detail values(1,'Disha','thane'),(2,'Chirantan','mumbai'),(3,'Rupali','Satara'),(4,'Tanaji','navi\_mumbai');**

**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.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/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 class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"* name=*"ds"*>

<property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"url"* value=*"jdbc:mysql://localhost:3306/springjdbc"* />

<property name=*"username"* value=*"root"*/>

<property name=*"password"* value=*"root"*/>

</bean>

<bean class=*"org.springframework.jdbc.core.JdbcTemplate"* name=*"jdbcTemplate"* p:dataSource

ref=*"ds"*/>

</beans>

**Main class**

**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( "kaizokuo ni ore wa naru!" );

ApplicationContext context = **new** ClassPathXmlApplicationContext("com/mca/config.xml");

JdbcTemplate temp = context.getBean("jdbcTemplate", JdbcTemplate.**class**);

// insert Query

String query1 = "insert into strawHat values(?,?,?)";

String query2 = "update strawHat set bounty=? where id=?";

String query3 = "delete from strawHat where id=?";

// fire query

**int** result1 = temp.update(query1,2,"zoro","1.2 Billion");

System.***out***.println("Number of records insetred " + result1);

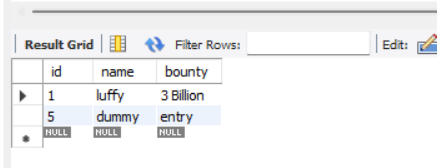
**int** result2 = temp.update(query2,"4 billion",1);

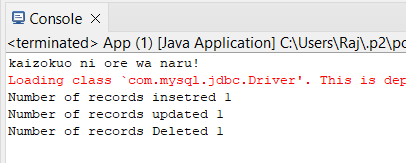
System.***out***.println("Number of records updated " + result2);

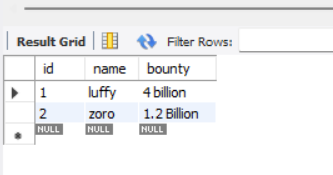
**int** result3 = temp.update(query3,5);

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

**Output:**







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

**Main class**

**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( "kaizokuo ni ore wa naru!" );

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** PreparedStatement createPreparedStatement(Connection con) **throws**

SQLException {

PreparedStatement ps = con.prepareStatement(query1);

ps.setInt(1, 3);

ps.setString(2, "zoro");

ps.setString(3, "1.1 Billion");

**return** ps;

}

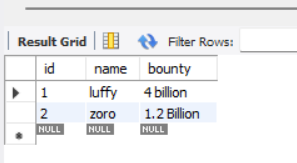
});

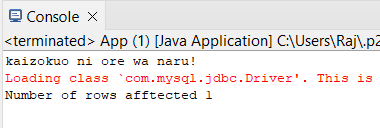
System.***out***.println("Number of rows afftected " + result);

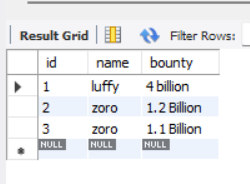
}

}

**Output:**







**PRACTICAL 10**

**ASSIGNMENT BASED SPRING BOOT AND RESTFUL WEB SERVICES**

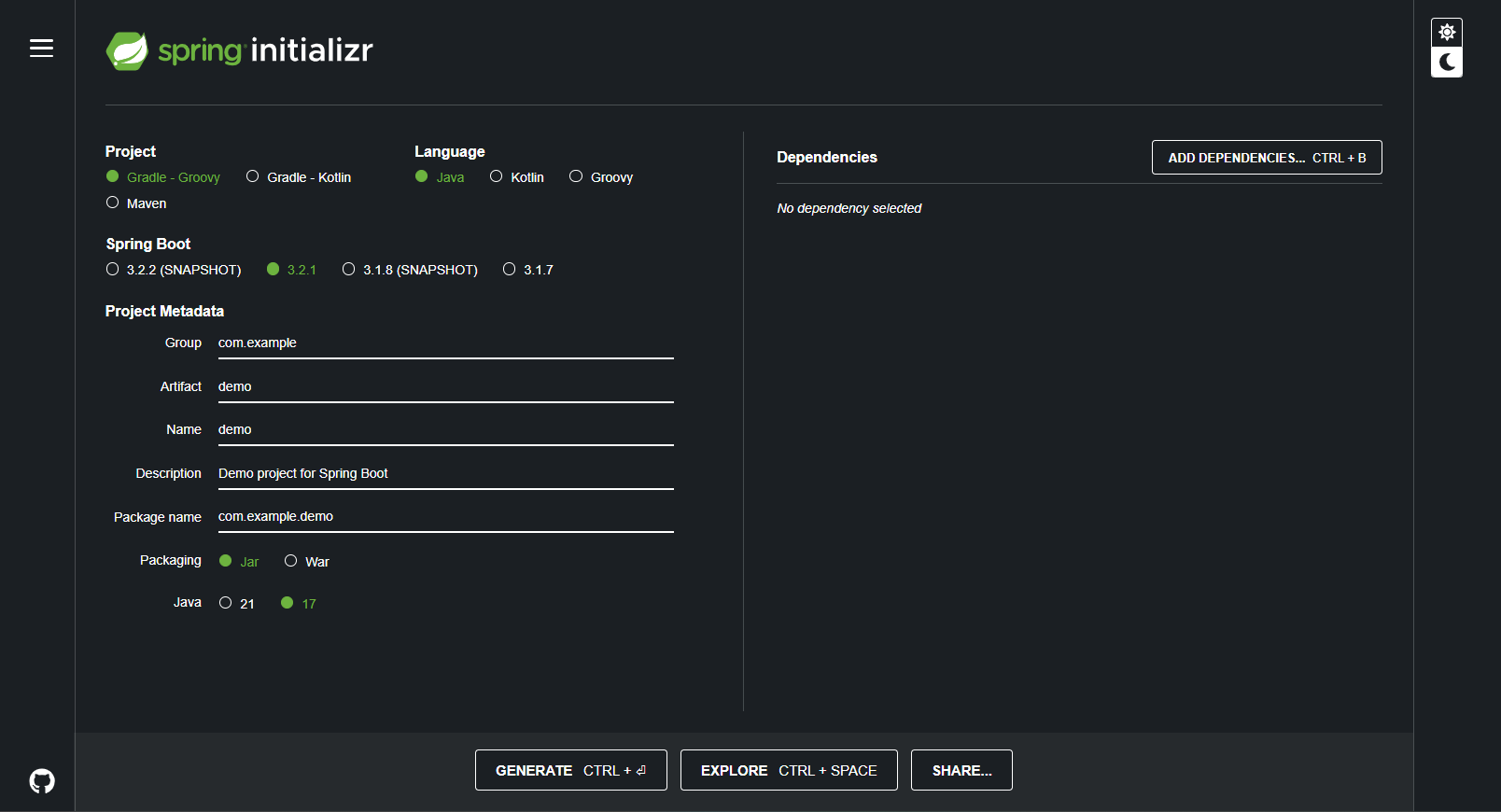
**1.Write a program to create a simple Spring Boot application prints a message**

**1) Go to Spring Initializr. Select the type of project (Maven).**

**Choose the language (Java). Select the Spring Boot version.**

**Fill in the project metadata. Add the necessary dependencies (at least spring-boot**

**starter-web). Click on “Generate” to download the project.**

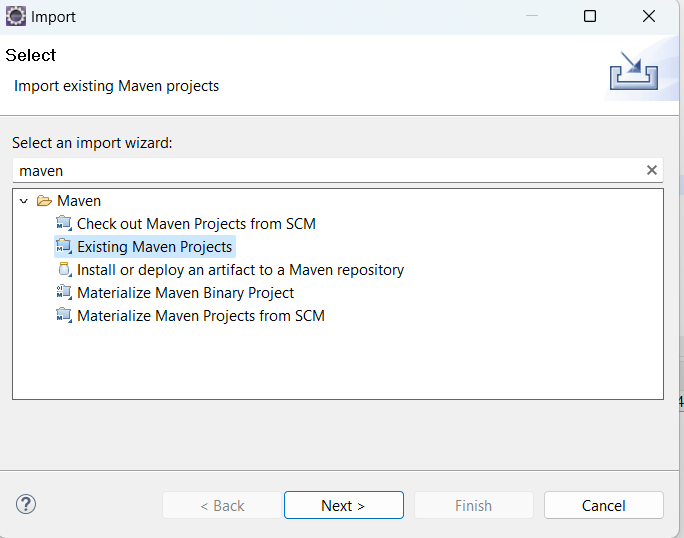


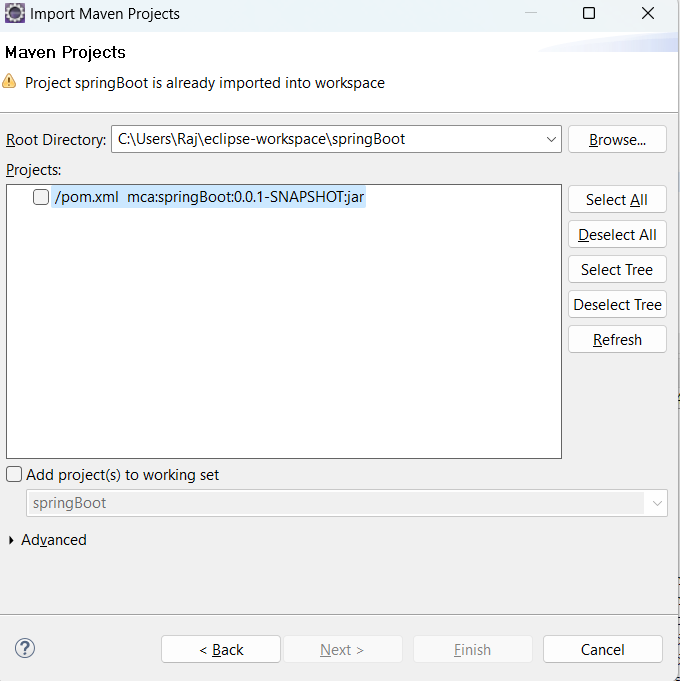
**2) Open Eclipse IDE. Navigate to File > Import.**

**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”.**





**Main class**

**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** main(String[] args) {

SpringApplication.run(myApplication.**class**, args);

}

@RestController

**public class** controller {

@GetMapping("/")

**public** String quote() {

**return** "Hero? No! We’re pirates! I love heroes, but I don’t wanna

be one!";

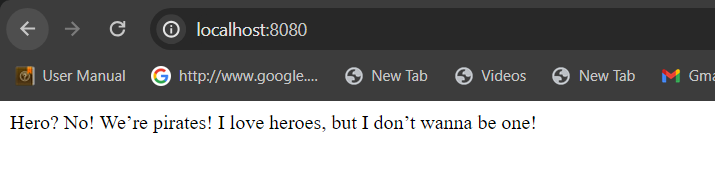
}

}

}

**Output :**





**Practical Exam Questions:**

Q 1.Write a program in Java using spring to accept two numbers and display Fibonacci series. Make use of setter dependency injection

Code:-

Step 1:-Create a Spring Configuration File (applicationContext.xml):

<!-- 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"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

<!-- Define a bean for the FibonacciCalculator class -->

<bean id="fibonacciCalculator" class="com.example.FibonacciCalculator">

<!-- Inject the dependencies using setter injection -->

<property name="firstNumber" value="10" />

<property name="secondNumber" value="20" />

</bean>

</beans>

Step 2:-

Create the FibonacciCalculator class:

// FibonacciCalculator.java

public class FibonacciCalculator {

private int firstNumber;

private int secondNumber;

// Setter methods for dependency injection

public void setFirstNumber(int firstNumber) {

this.firstNumber = firstNumber;

}

public void setSecondNumber(int secondNumber) {

this.secondNumber = secondNumber;

}

// Method to calculate and display Fibonacci series

public void displayFibonacciSeries() {

System.out.println("Fibonacci Series:");

int n = 10; // Change the value as per your requirement

int firstTerm = firstNumber;

int secondTerm = secondNumber;

for (int i = 1; i <= n; ++i) {

System.out.print(firstTerm + ", ");

int nextTerm = firstTerm + secondTerm;

firstTerm = secondTerm;

secondTerm = nextTerm;

}

}

}

Step 3:Create the main application class with Spring context:

// MainApp.java

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

// Load the Spring context

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

// Retrieve the FibonacciCalculator bean from the context

FibonacciCalculator fibonacciCalculator = (FibonacciCalculator) context.getBean("fibonacciCalculator");

// Call the method to display Fibonacci series

fibonacciCalculator.displayFibonacciSeries();

}

}

Q.2 Write a java program to accept a number and display if it is a prime number using lambda expression.

Code:-

package prac;

import java.util.Scanner;

import java.util.function.Predicate;

public class prac\_1 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Accept a number from the user

System.out.print("Enter a number: ");

int numberToCheck = scanner.nextInt();

// Use a lambda expression to check if the number is prime

Predicate<Integer> isPrime = n -> {

if (n <= 1) {

return false;

}

for (int i = 2; i <= Math.sqrt(n); i++) {

if (n % i == 0) {

return false;

}

}

return true;

};

// Display the result

if (isPrime.test(numberToCheck)) {

System.out.println(numberToCheck + " is a prime number.");

} else {

System.out.println(numberToCheck + " is not a prime number.");

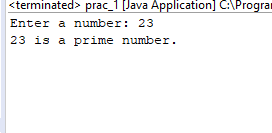
}

scanner.close();

}

}

O/p:-



Q.2 Devise an Interest Calculator that accepts Principal amount, number of years and rate of interest from user. The calculator then asks for option as Simple Interest or Compound Interest. The interest should be displayed as per the choice. Design a webpage using JSP to demonstrate the functioning of interest calculator. Make necessary assumptions wherever needed.

Code:-

1.Interest Calculator Servlet (InterestCalculatorServlet.java):

java code-

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/InterestCalculatorServlet")

public class InterestCalculatorServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Retrieve user inputs

double principal = Double.parseDouble(request.getParameter("principal"));

int years = Integer.parseInt(request.getParameter("years"));

double rate = Double.parseDouble(request.getParameter("rate"));

// Determine the type of interest (Simple or Compound)

String interestType = request.getParameter("interestType");

double interest = 0;

if (interestType.equals("simple")) {

interest = calculateSimpleInterest(principal, years, rate);

} else if (interestType.equals("compound")) {

interest = calculateCompoundInterest(principal, years, rate);

}

// Set the calculated interest in the request attribute

request.setAttribute("interest", interest);

// Forward the request to the JSP page

request.getRequestDispatcher("interestResult.jsp").forward(request, response);

}

private double calculateSimpleInterest(double principal, int years, double rate) {

return principal \* years \* rate / 100;

}

private double calculateCompoundInterest(double principal, int years, double rate) {

return principal \* Math.pow((1 + rate / 100), years) - principal;

}

}

JSP Page (interestCalculator.jsp):

jsp code

<!-- interestCalculator.jsp -->

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Interest Calculator</title>

</head>

<body>

<h2>Interest Calculator</h2>

<form action="InterestCalculatorServlet" method="post">

<label for="principal">Principal Amount:</label>

<input type="text" id="principal" name="principal" required><br>

<label for="years">Number of Years:</label>

<input type="text" id="years" name="years" required><br>

<label for="rate">Rate of Interest:</label>

<input type="text" id="rate" name="rate" required><br>

<label>Choose Interest Type:</label><br>

<input type="radio" id="simple" name="interestType" value="simple" checked>

<label for="simple">Simple Interest</label><br>

<input type="radio" id="compound" name="interestType" value="compound">

<label for="compound">Compound Interest</label><br>

<input type="submit" value="Calculate Interest">

</form>

</body>

</html>

JSP Result Page (interestResult.jsp):

jsp code-

<!-- interestResult.jsp -->

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Interest Result</title>

</head>

<body>

<h2>Interest Result</h2>

<p>Calculated Interest: <%= request.getAttribute("interest") %></p>

</body>

</html>

Q.2 Write a java program to accept two numbers - add them and multiply them using lambda expressions. Write different lambda expressions for each activity.

Code:

import java.util.Scanner;

import java.util.function.BinaryOperator;

public class CalculatorLambda {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Accept two numbers from the user

System.out.print("Enter the first number: ");

double num1 = scanner.nextDouble();

System.out.print("Enter the second number: ");

double num2 = scanner.nextDouble();

// Lambda expression for addition

BinaryOperator<Double> addFunction = (x, y) -> x + y;

// Lambda expression for multiplication

BinaryOperator<Double> multiplyFunction = (x, y) -> x \* y;

// Perform addition and multiplication using the lambda expressions

double sum = addFunction.apply(num1, num2);

double product = multiplyFunction.apply(num1, num2);

// Display the results

System.out.println("Sum: " + sum);

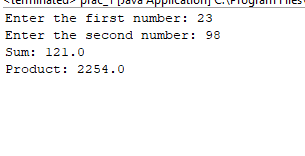
System.out.println("Product: " + product);

scanner.close();

}

}

O/p:



Q3 Write a Java program in eclipse using lambda expression with three parameters to display the largest of these numbers.

Code:-

import java.util.Scanner;

import java.util.function.Function;

public class LargestNumberLambda {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Accept three numbers from the user

System.out.print("Enter the first number: ");

double num1 = scanner.nextDouble();

System.out.print("Enter the second number: ");

double num2 = scanner.nextDouble();

System.out.print("Enter the third number: ");

double num3 = scanner.nextDouble();

// Lambda expression to find the largest number

Function<Double, Function<Double, Function<Double, Double>>> findLargest = x -> y -> z -> {

double max = x > y ? x : y;

return max > z ? max : z;

};

// Use the lambda expression to find the largest number

double largestNumber = findLargest.apply(num1).apply(num2).apply(num3);

// Display the result

System.out.println("The largest number is: " + largestNumber);

scanner.close();

}

}

Q.5 Design a web page using JSP for a COVID vaccination center to accept

bookings for vaccination for a day. Design a second webpage to display

accepted names, their token numbers and the time for vaccination on that day

in a table. Both the pages must be interlinked. Make necessary assumptions

wherever needed

ChatGPT

Certainly! Below is a simple example of a web application using JSP for a COVID vaccination center to accept bookings and display accepted names, their token numbers, and vaccination time.

Booking Page (booking.jsp):

jsp

Copy code

<!-- booking.jsp -->

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>COVID Vaccination Booking</title>

</head>

<body>

<h2>COVID Vaccination Booking</h2>

<form action="BookingServlet" method="post">

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

<input type="text" id="name" name="name" required><br>

<label for="token">Token Number:</label>

<input type="text" id="token" name="token" required><br>

<label for="time">Vaccination Time:</label>

<input type="text" id="time" name="time" required><br>

<input type="submit" value="Book Vaccination">

</form>

</body>

</html>

Booking Servlet (BookingServlet.java):

java code

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/BookingServlet")

public class BookingServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

private static final List<Booking> bookings = new ArrayList<>();

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

// Retrieve user inputs

String name = request.getParameter("name");

int token = Integer.parseInt(request.getParameter("token"));

String time = request.getParameter("time");

// Create a new booking

Booking newBooking = new Booking(name, token, time);

bookings.add(newBooking);

// Set the booked vaccinations in the request attribute

request.setAttribute("bookings", bookings);

// Forward the request to the display page

request.getRequestDispatcher("displayBookings.jsp").forward(request, response);

}

}

Booking Class (Booking.java):

java

Copy code

public class Booking {

private String name;

private int token;

private String time;

public Booking(String name, int token, String time) {

this.name = name;

this.token = token;

this.time = time;

}

// Getters (no setters for simplicity)

public String getName() {

return name;

}

public int getToken() {

return token;

}

public String getTime() {

return time;

}

}

Display Bookings Page (displayBookings.jsp):

jsp

Copy code

<!-- displayBookings.jsp -->

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

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

<%@ page import="Booking" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Booked Vaccinations</title>

</head>

<body>

<h2>Booked Vaccinations</h2>

<table border="1">

<thead>

<tr>

<th>Name</th>

<th>Token Number</th>

<th>Vaccination Time</th>

</tr>

</thead>

<tbody>

<%

List<Booking> bookings = (List<Booking>) request.getAttribute("bookings");

for (Booking booking : bookings) {

%>

<tr>

<td><%= booking.getName() %></td>

<td><%= booking.getToken() %></td>

<td><%= booking.getTime() %></td>

</tr>

<%

}

%>

</tbody>

</table>

</body>

</html>

Q.4 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.

import java.util.ArrayList;

import java.util.List;

public class ListExample {

public static void main(String[] args) {

// Create a List of Strings

List<String> stringList = new ArrayList<>();

// Add items to the list

stringList.add("Apple");

stringList.add("Banana");

stringList.add("Orange");

stringList.add("Grapes");

stringList.add("Mango");

// Use a for-each loop to print each item in the list

System.out.println("Items in the list:");

for (String item : stringList) {

System.out.println(item);

}

}

}

Q 6 Design a webpage using Java Server Pages to accept three numbers from user and display its average. Make use of use of JSP declaration, scriptlet and expression in your web page.

<%@ page language="java" contentType="text/html; charset=UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Calculate Average</title>

</head>

<body>

<h2>Calculate Average</h2>

<%

// JSP declaration

double num1, num2, num3, average;

%>

<form action="" method="post">

<label for="num1">Enter Number 1:</label>

<input type="text" name="num1" id="num1" required><br>

<label for="num2">Enter Number 2:</label>

<input type="text" name="num2" id="num2" required><br>

<label for="num3">Enter Number 3:</label>

<input type="text" name="num3" id="num3" required><br>

<input type="submit" value="Calculate Average">

</form>

<%

// JSP scriptlet

if (request.getMethod().equalsIgnoreCase("post")) {

try {

num1 = Double.parseDouble(request.getParameter("num1"));

num2 = Double.parseDouble(request.getParameter("num2"));

num3 = Double.parseDouble(request.getParameter("num3"));

average = (num1 + num2 + num3) / 3.0;

%>

<p>The average of <%= num1 %>, <%= num2 %>, and <%= num3 %> is: <%= average %></p>

<%

} catch (NumberFormatException e) {

%>

<p>Please enter valid numeric values for all three numbers.</p>

<%

}

}

%>

</body>

</html>

Q. Write a program in Java using spring to accept name of a state of our country and display if assembly elections are to be held in the year 2021 or not. Make use of constructor dependency injection. Make necessary assumptions.

StateElectionChecker.java:

java

Copy code

public class StateElectionChecker {

private String stateName;

// Constructor for dependency injection

public StateElectionChecker(String stateName) {

this.stateName = stateName;

}

public void checkElectionStatus() {

// Assume some logic to determine election status (e.g., based on state-specific rules)

boolean electionsIn2021 = /\* Logic to determine if elections are in 2021 \*/ true;

if (electionsIn2021) {

System.out.println("Assembly elections are to be held in " + stateName + " in the year 2021.");

} else {

System.out.println("No assembly elections are scheduled in " + stateName + " for the year 2021.");

}

}

}

SpringConfig.xml:

Create a Spring configuration XML file named SpringConfig.xml to define the Spring beans.

xml

Copy code

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

<!-- Define the StateElectionChecker bean with constructor injection -->

<bean id="stateElectionChecker" class="StateElectionChecker">

<constructor-arg value="Assam" /> <!-- Provide a default state name or use setter injection -->

</bean>

</beans>

MainApp.java:

Create a main application to load the Spring context and execute the election status check.

java

Copy code

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

// Load the Spring context

ApplicationContext context = new ClassPathXmlApplicationContext("SpringConfig.xml");

// Retrieve the StateElectionChecker bean

StateElectionChecker electionChecker = (StateElectionChecker) context.getBean("stateElectionChecker");

// Execute the election status check

electionChecker.checkElectionStatus();

}

}

Q. Define a generic class called GenT, and a second class called GenDemo, which uses GenT. The GenT class should have a parameter T that will be replaced by a data type whenever the object of type GenT is created. Write a Java Program to implement simple use of Generic class and demonstrate by using various primitive data types in place of T.

// Generic class definition

class GenT<T> {

private T value;

// Constructor

public GenT(T value) {

this.value = value;

}

// Getter and setter methods

public T getValue() {

return value;

}

public void setValue(T value) {

this.value = value;

}

}

// Class using GenT

class GenDemo {

public static void main(String[] args) {

// Create GenT objects with different data types

GenT<Integer> intGen = new GenT<>(42);

GenT<Double> doubleGen = new GenT<>(3.14);

GenT<Boolean> booleanGen = new GenT<>(true);

// Display values

System.out.println("Integer Value: " + intGen.getValue());

System.out.println("Double Value: " + doubleGen.getValue());

System.out.println("Boolean Value: " + booleanGen.getValue());

// Update values

intGen.setValue(99);

doubleGen.setValue(2.718);

booleanGen.setValue(false);

// Display updated values

System.out.println("Updated Integer Value: " + intGen.getValue());

System.out.println("Updated Double Value: " + doubleGen.getValue());

System.out.println("Updated Boolean Value: " + booleanGen.getValue());

}

}

Q.5 Write a Java program using lambda expression to calculate the sum of numbers from 1 to 100. Also write another lambda expression to display numbers from 1 to 100 in reverse order.

import java.util.stream.IntStream;

public class LambdaExpressionsExample {

public static void main(String[] args) {

// Lambda expression to calculate the sum of numbers from 1 to 100

IntStream numbers = IntStream.rangeClosed(1, 100);

int sum = numbers.reduce(0, (x, y) -> x + y);

System.out.println("Sum of numbers from 1 to 100: " + sum);

// Lambda expression to display numbers from 1 to 100 in reverse order

IntStream reversedNumbers = IntStream.rangeClosed(1, 100)

.map(i -> 100 - i + 1);

System.out.println("Numbers from 1 to 100 in reverse order:");

reversedNumbers.forEach(System.out::println);

}

}

Q9 rite a program in Java using spring to accept a string and a number and

display the string those many numbers of times. Make use of setter

dependency injection.

MessagePrinter.java (Spring Bean):

java

Copy code

public class MessagePrinter {

private String message;

private int repeatCount;

// Setter method for message injection

public void setMessage(String message) {

this.message = message;

}

// Setter method for repeatCount injection

public void setRepeatCount(int repeatCount) {

this.repeatCount = repeatCount;

}

public void printMessage() {

for (int i = 0; i < repeatCount; i++) {

System.out.println(message);

}

}

}

SpringConfig.xml:

Create a Spring configuration XML file named SpringConfig.xml to define the Spring beans and inject dependencies.

xml

Copy code

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

<!-- Define the MessagePrinter bean with setter injection -->

<bean id="messagePrinter" class="MessagePrinter">

<property name="message" value="Hello, Spring!" />

<property name="repeatCount" value="5" /> <!-- Set the desired repeat count -->

</bean>

</beans>

MainApp.java:

Create a main application to load the Spring context and execute the printMessage method.

java

Copy code

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

// Load the Spring context

ApplicationContext context = new ClassPathXmlApplicationContext("SpringConfig.xml");

// Retrieve the MessagePrinter bean

MessagePrinter messagePrinter = (MessagePrinter) context.getBean("messagePrinter");

// Execute the printMessage method

messagePrinter.printMessage();

}

}

Q 7

Write a program in Java using spring to accept name and marks for five subjects. Display the marklist of that student. Make use of constructor dependency injection.

StudentMarkList.java (Spring Bean):

java

Copy code

public class StudentMarkList {

private String studentName;

private int[] marks;

// Constructor for constructor injection

public StudentMarkList(String studentName, int[] marks) {

this.studentName = studentName;

this.marks = marks;

}

public void displayMarkList() {

System.out.println("Marklist for " + studentName + ":");

for (int i = 0; i < marks.length; i++) {

System.out.println("Subject " + (i + 1) + ": " + marks[i] + " marks");

}

}

}

SpringConfig.xml:

Create a Spring configuration XML file named SpringConfig.xml to define the Spring beans and inject dependencies.

xml

Copy code

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

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

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

<!-- Define the StudentMarkList bean with constructor injection -->

<bean id="studentMarkList" class="StudentMarkList">

<constructor-arg value="John Doe" /> <!-- Set the student's name -->

<constructor-arg>

<list>

<value>90</value> <!-- Marks for Subject 1 -->

<value>85</value> <!-- Marks for Subject 2 -->

<value>92</value> <!-- Marks for Subject 3 -->

<value>88</value> <!-- Marks for Subject 4 -->

<value>95</value> <!-- Marks for Subject 5 -->

</list>

</constructor-arg>

</bean>

</beans>

MainApp.java:

Create a main application to load the Spring context and execute the displayMarkList method.

java

Copy code

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

// Load the Spring context

ApplicationContext context = new ClassPathXmlApplicationContext("SpringConfig.xml");

// Retrieve the StudentMarkList bean

StudentMarkList studentMarkList = (StudentMarkList) context.getBean("studentMarkList");

// Execute the displayMarkList method

studentMarkList.displayMarkList();

}

}

Q 6 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.

import java.util.ArrayList;

import java.util.List;

public class ListExample {

public static void main(String[] args) {

// Create a List of Strings

List<String> stringList = new ArrayList<>();

// Add items to the list

stringList.add("Apple");

stringList.add("Banana");

stringList.add("Orange");

stringList.add("Grapes");

stringList.add("Mango");

// Use a for-each loop to print each item in the list

System.out.println("Items in the list:");

for (String item : stringList) {

System.out.println(item);

}

}

}

Q esign a webpage using JSP to show the working of page directive and include directive. The page directive must have atleast 4 attributes. The include directive must include a header page at the top and a footer page at the bottom of the JSP web page.

ndex.jsp:

jsp

Copy code

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"

import="java.util.Date"

session="false"

buffer="8kb" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>JSP Page with Directives</title>

</head>

<body>

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

<h2>Welcome to JSP Page with Directives</h2>

<p>Current Date and Time: <%= new Date() %></p>

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

</body>

</html>

header.jsp:

jsp

Copy code

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8" %>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Header</title>

</head>

<body>

<header>

<h1>This is the Header</h1>

<hr>

</header>

footer.jsp:

jsp

Copy code

</body>

</html>

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8" %>

<footer>

<hr>

<p>&copy; 2022 Your Company</p>

</footer>

Q Define a generic class called GenAB, and a second class called GenDemo, which uses GenAB. The GenAB class should have atleast two parameters namely A and B that will be replaced by different data types whenever the object of type GenAB is created. Write a Java program to implement simple use of Generic class and demonstrate by using various primitive data types in place of generic types

// Generic class definition

class GenAB<A, B> {

private A valueA;

private B valueB;

// Constructor

public GenAB(A valueA, B valueB) {

this.valueA = valueA;

this.valueB = valueB;

}

// Getter methods

public A getValueA() {

return valueA;

}

public B getValueB() {

return valueB;

}

}

// Class using GenAB

class GenDemo {

public static void main(String[] args) {

// Create GenAB objects with different data types

GenAB<Integer, String> intStringGen = new GenAB<>(42, "Java");

GenAB<Double, Character> doubleCharGen = new GenAB<>(3.14, 'G');

GenAB<Boolean, Long> booleanLongGen = new GenAB<>(true, 987654321L);

// Display values

System.out.println("Integer and String Values: " + intStringGen.getValueA() + ", " + intStringGen.getValueB());

System.out.println("Double and Character Values: " + doubleCharGen.getValueA() + ", " + doubleCharGen.getValueB());

System.out.println("Boolean and Long Values: " + booleanLongGen.getValueA() + ", " + booleanLongGen.getValueB());

}

}