# PRACTICAL 1 :ASSIGNMENTS ON JAVA GENERICS

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

**package** Genricsclass;

**class** Main1<T> {

T obj;

**void** add(T obj) {

**this**.obj=obj;

}

T get() {

**return** obj;

}m

}

**public** **class** Genricsclass {

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

Main1<Integer> nn =**new** Main1<Integer>();

nn.add(87);

Main1<String> ss =**new** Main1<String>();

ss.add("Sagar D Londhe");

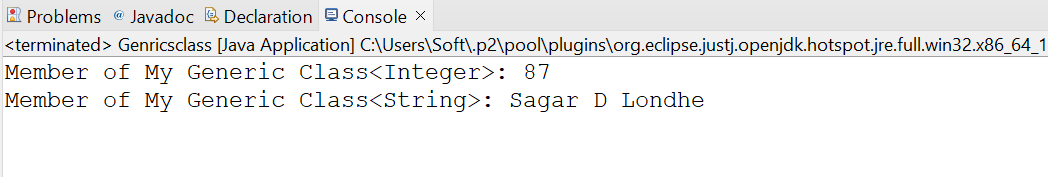
System.***out***.println("Member of My Generic Class<Integer>: "+ nn.get());

System.***out***.println("Member of My Generic Class<String>: "+ ss.get());

}

}

## Output:



### Write a Java Program to demonstrate Generic Methods.

**package** GenericMethod;

**public** **class** GenericMethod {

**public** **static** < E > **void** printArray( E[] inputArray ) {

// Display array elements

**for**(E element : inputArray) { System.***out***.printf("%s ", element);

}

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

}

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

Integer[] intArray = { 1, 2, 3, 4, 5 };

Double[] doubleArray = { 1.1, 2.2, 3.3, 4.4 };

Character[] charArray = { 'S', 'A', 'G', 'A', 'R' };

// integer Array

System.***out***.println("Array integerArray contains:");

*printArray*(intArray);

// double array

System.***out***.println("\nArray doubleArray contains:");

*printArray*(doubleArray);

// Charter Array

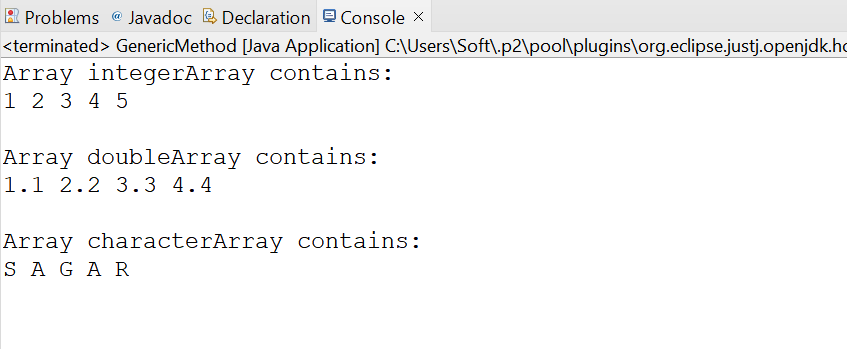
System.***out***.println("\nArray characterArray contains:");

*printArray*(charArray);

}

}

## Output:



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

**package** Wildcard;

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

**public** **class** Wildcards {

**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));

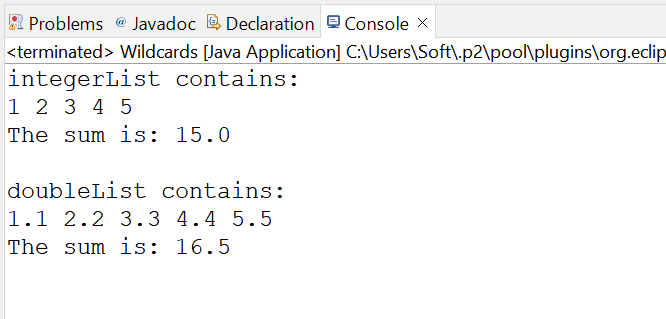
List<Double> doubleList = Arrays.*asList*(1.1, 2.2, 3.3, 4.4, 5.5); System.***out***.println("\ndoubleList contains: "); *printList*(doubleList);

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

}

}

**Output:**



# PRACTICAL 2 :ASSIGNMENTS ON LIST INTERFACE

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

**package** Listinterface;

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

**public** **class** ListInterface {

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

List<String> items = Arrays.*asList*("Sagar", "Prathamesh", "Vishal", "Rahul", "Sai");

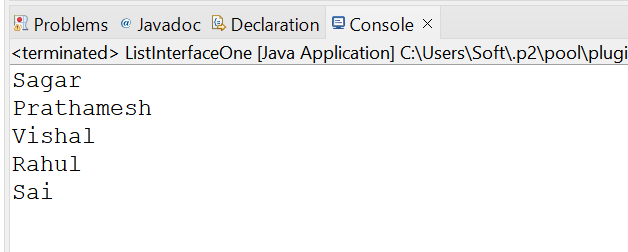
**for** (String item : items) { System.***out***.println(item);

}

}

}

## Output:



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

**package** listinterface;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.ListIterator;

**public** **class** listiterator {

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

// Create a List containing items

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

itemList.add("Item 1");

itemList.add("Item 2");

itemList.add("Item 3");

itemList.add("Item 4");

itemList.add("Item 5");

// Print items in forward direction using ListIterator

System.***out***.println("Forward direction:");

ListIterator<String> forwardIterator = itemList.listIterator();

**while** (forwardIterator.hasNext()) {

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

}

// Print items in reverse direction using ListIterator

System.***out***.println("\nReverse direction:");

ListIterator<String> reverseIterator = itemList.listIterator(itemList.size());

**while** (reverseIterator.hasPrevious()) {

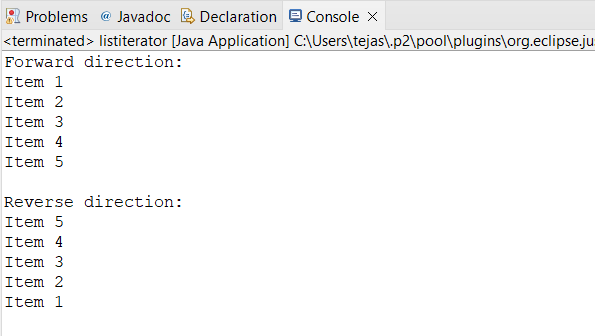
System.***out***.println(reverseIterator.previous());

}

}

}

**Output:**

****

# PRACTICAL 3: ASSIGNMENTS ON SET INTERFACE

### 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** setinterface;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.util.LinkedList;

**import** java.util.List;

**import** java.util.Set;

**import** java.util.ListIterator;

**public** **class** setiterator {

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

// Create a Set containing items

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

itemSet.add("a");

itemSet.add("b");

itemSet.add("c");

itemSet.add("d");

itemSet.add("e");

// Print items in forward direction using Iterator

System.***out***.println("Forward direction using set iterator:");

Iterator<String> forwardIterator = itemSet.iterator();

**while** (forwardIterator.hasNext()) {

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

}

// Print items in reverse direction using ListIterator

System.***out***.println("\nReverse direction using set iterator:");

List<String> itemList = **new** LinkedList<>(itemSet); // Convert Set to List

ListIterator<String> reverseIterator = itemList.listIterator(itemList.size());

**while** (reverseIterator.hasPrevious()) {

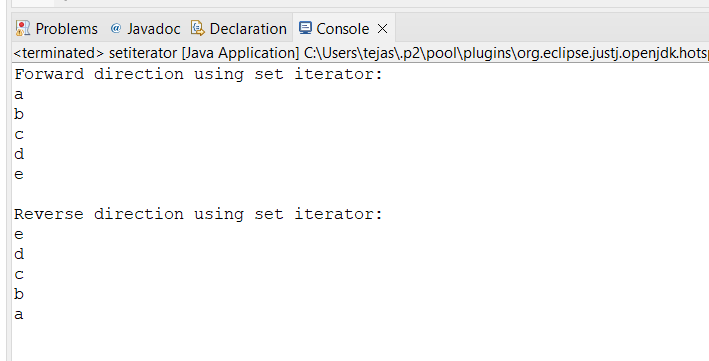
System.***out***.println(reverseIterator.previous());

}

}

}

## Output:

****

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

* 1. **Add items in the set.**
  2. **Insert items of one set in to other set.**
  3. **Remove items from the set**
  4. **Search the specified item in the set**

**package** setinterface;

**import** java.util.HashSet;

**import** java.util.Set;

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

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

// Creating two sets

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

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

// a. Add items to the set

set1.add("Item 1");

set1.add("Item 2");

set1.add("Item 3");

System.***out***.println("Set 1: " + set1);

// b. Insert items of set1 into set2

set2.addAll(set1);

System.***out***.println("Set 2 after inserting items from Set 1: " + set2);

// c. Remove items from set2

set2.remove("Item 2");

System.***out***.println("Set 2 after removing 'Item 2': " + set2);

// d. Search the specified item in set1

String searchItem = "Item 3";

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

System.***out***.println("'" + searchItem + "' found in Set 1.");

} **else** {

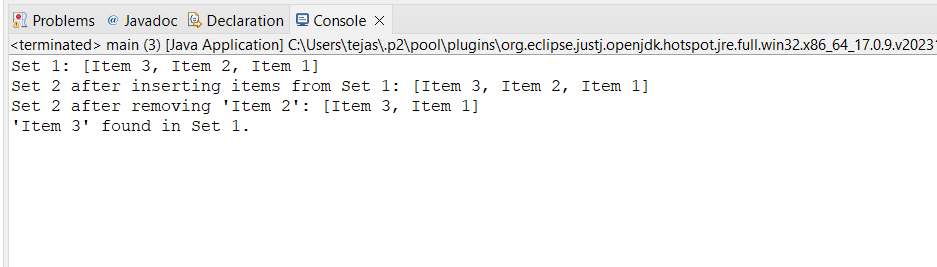
System.***out***.println("'" + searchItem + "' not found in Set 1.");

}

}

}

**Output:**

****

# PRACTICAL 4 : 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:

1. **Add items in the map.**
2. **Remove items from the map**
3. **Search specific key from the map**
4. **Get value of the specified key**
5. **Insert map elements of one map in to other map.**
6. **Print all keys and values of the map**

**package** mapinterface;

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

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

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

// Creating maps

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

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

// a. Add items in the map

map1.put(1, "Rose");

map1.put(2, "Lily");

map1.put(3, "Marigold");

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

System.***out***.println("Map 1: " + map1);

// b. Remove items from the map

map1.remove(3);

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

System.***out***.println("Map 1 after removal: " + map1);

// c. Search specific key from the map

**int** keyToSearch = 1;

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

System.***out***.println("Key " + keyToSearch + " found in Map 1.");

} **else** {

System.***out***.println("Key " + keyToSearch + " not found in Map 1.");

}

// d. Get value of the specified key

**int** specifiedKey = 3;

String value = map1.get(specifiedKey);

System.***out***.println("Value associated with key " + specifiedKey + ": " + value);

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

map2.putAll(map1);

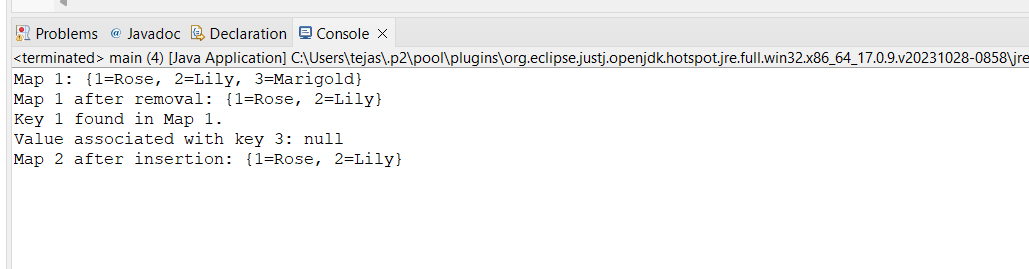
// f. Print all keys and values of the second map

System.***out***.println("Map 2 after insertion: " + map2);

}

}

**Output:**

****

# PRACTICAL 5: ASSIGNMENTS ON LAMBDA EXPRESSION

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

**package** LAMBDAEXPRESSION;

**public** **class** LambdaA {

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

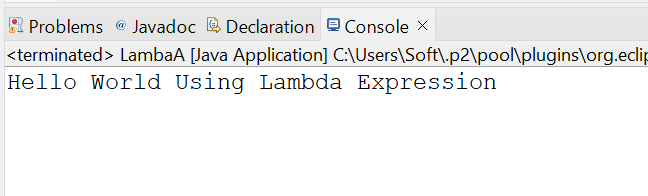
Runnable helloWorld = () -> System.***out***.println("Hello World Using Lambda Expression");

helloWorld.run();

}

}

## Output:



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

**package** LAMBDAEXPRESSION;

**interface** MessagePrinter {

**void** print(String message);

}

**public** **class** LambdaB {

**public** **static** **void** main(String[] args) { MessagePrinter printMessage = (message) ->

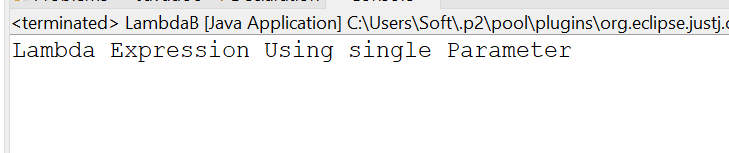
System.***out***.println(message);

printMessage.print("Lambda Expression Using single Parameter");

}

}

## Output:



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

**package** LAMBDAEXPRESSION;

**interface** Adder {

**int** add(**int** num1, **int** num2);

}

**public** **class** LambdaCAddTwoNo {

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

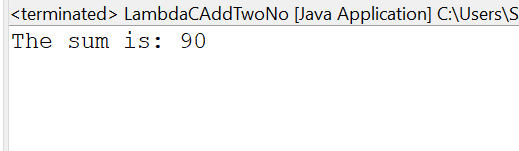
Adder addTwoNumbers = (num1, num2) -> num1 + num2;

**int** sum = addTwoNumbers.add(87, 3); System.***out***.println("The sum is: " + sum);

}

}

## Output:



### 

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

* 1. **Convert Fahrenheit to Celcius**
  2. **Convert Kilometers to Miles.**

**package** LAMBDAEXPRESSION;

@FunctionalInterface

**interface** Converter {

**double** convert(**double** input);

}

**public** **class** LambdaDConvert {

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

Converter fahrenheitToCelsius = (fahrenheit) -> (fahrenheit - 32) \* 5.0/9.0;

Converter kilometersToMiles = (kilometers) -> kilometers \*

0.621371;

// Execute the lambda expressions

**double** celsius = fahrenheitToCelsius.convert(100);

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

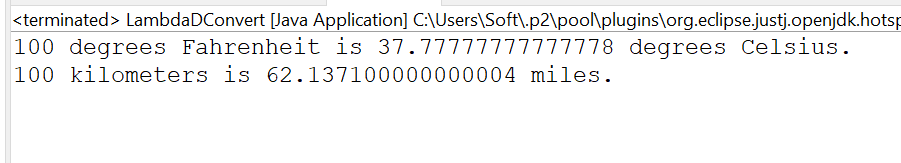
**double** miles = kilometersToMiles.convert(100);

System.***out***.println("100 kilometers is " + miles + " miles.");

}

}

## Output:



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

**package** LAMBDAEXPRESSION;

@FunctionalInterface **interface** Adder2 {

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

}

@FunctionalInterface

**interface** Subtractor {

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

}

**public** **class** LambdaEReturnKeyword {

**public** **static** **void** main(String[] args) { Adder2 adder = (a, b) -> a + b;

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

**int** sum = adder.add(87, 9);

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

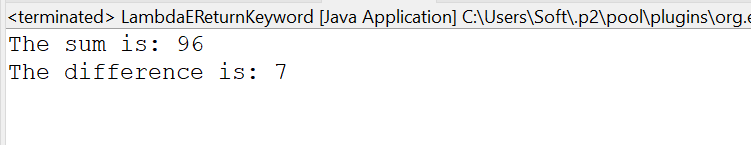
**int** difference = subtractor.subtract(87, 80);

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

}

}

## Output:



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

**package** LAMBDAEXPRESSION;

@FunctionalInterface

**interface** ConcatenateInterface {

String concatenate(String a, String b);

}

**public** **class** LambdaFconcat {

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

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

String str1 = "Sagar Londhe";

String str2 = " Lambda Expression to concatenate two strings";

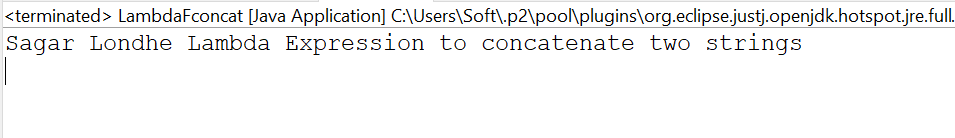
String result = concat.concatenate(str1, str2);

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

}

}

**Output :**



# PRACTICAL 6: ASSIGNMENTS BASED ON WEB APPLICATION DEVELOPMENT USING JSP

### Write Programs to demonstrate different Implicit Objects

* 1. **OUT**
  2. **Request**
  3. **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 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");

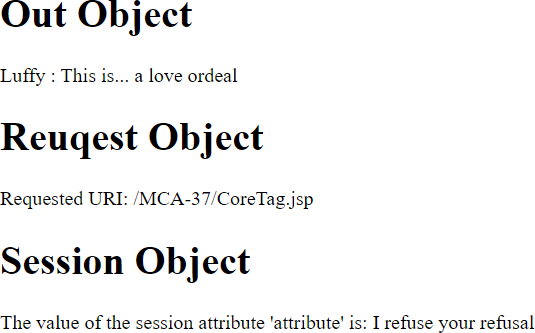
out.println("The value of the session attribute 'attribute' is: " + attribute);

%>

</body>

</html>

## Output :



### Write Programs 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:



### 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>

<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 :



### 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"](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>

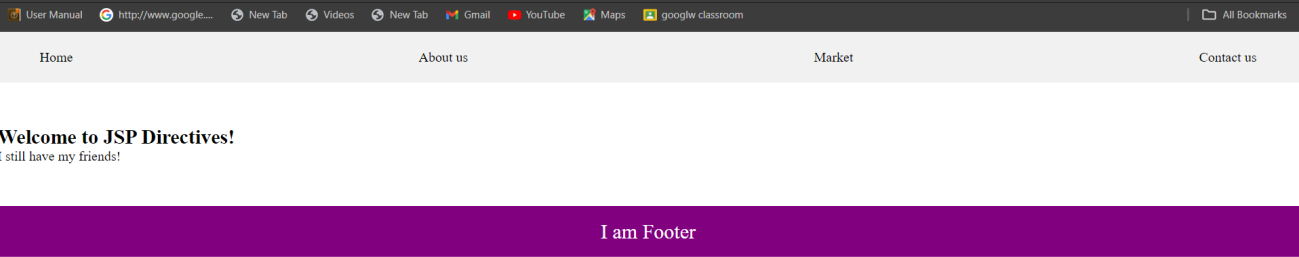
<c:out value=*"*${'I still have my friends!'}*"*/>

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

</body>

</html>

## Output :



### 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")

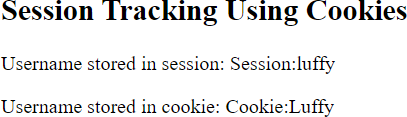
%></p>

<p>Username stored in cookie: <%= usernameCookie.getValue() %></p>

</body>

</html>

## Output :



### 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:forEachvar=*"i"*begin=*"1"*end=*"5"*>

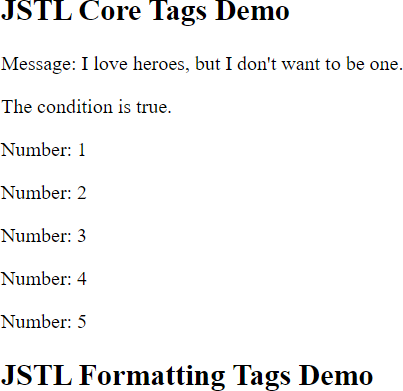
<p>Number: ${i}</p>

</c:forEach>

</body>

</html>

## Output :



### 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"](http://java.sun.com/jsp/jstl/core)* prefix = *"c"*%>

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

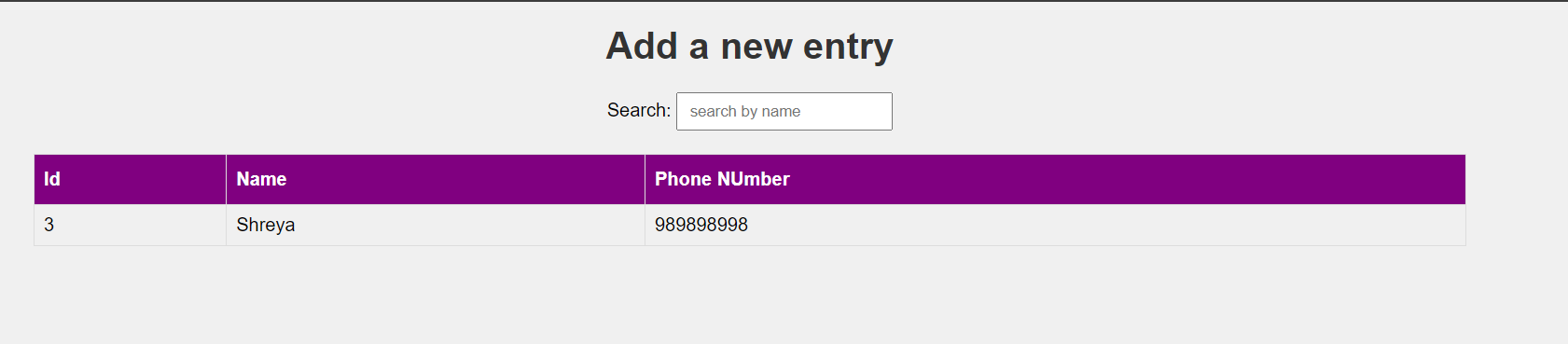
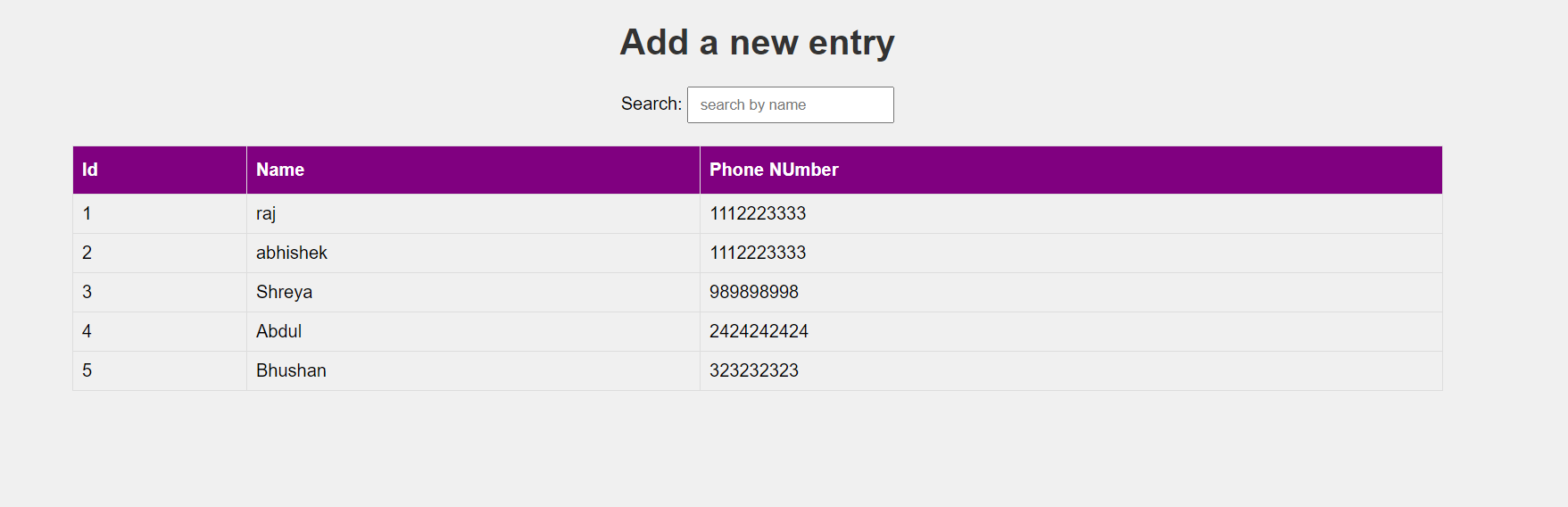
</table>

</body>

</body>

</html>

## Output :



### 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"*>

<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>

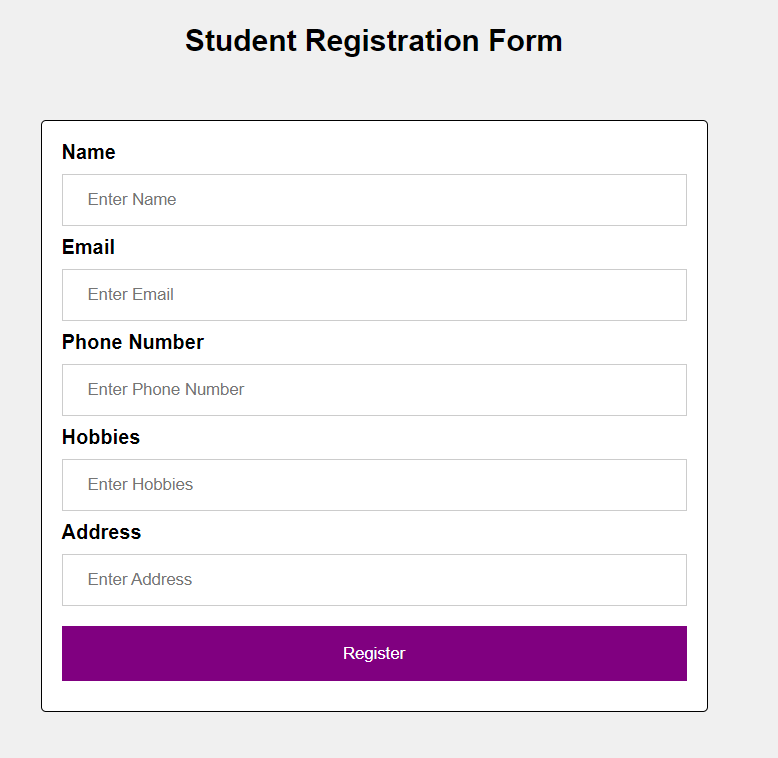
<button type=*"submit"*>Register</button>

</div>

</body>

</html>

**Output :**

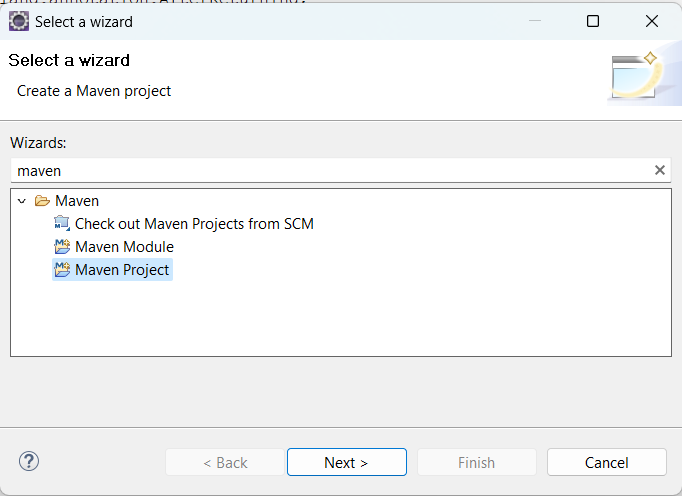


# PRACTICAL 7: ASSIGNMENT BASED SPRING FRAMEWORK

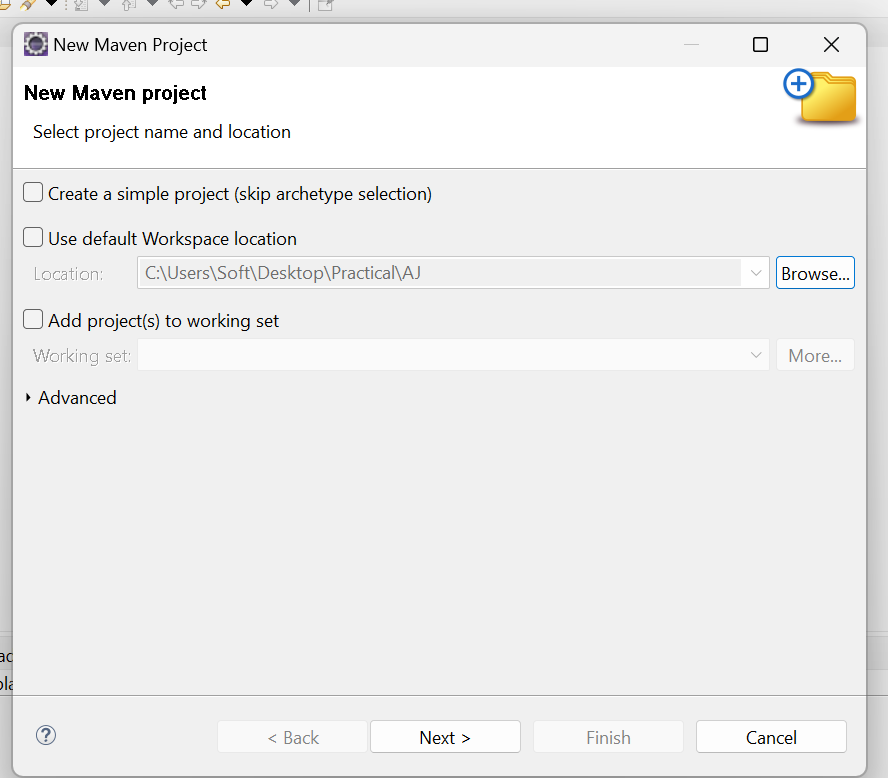
### Write a program to print Singer Name and Age using spring framework.

## Maven project

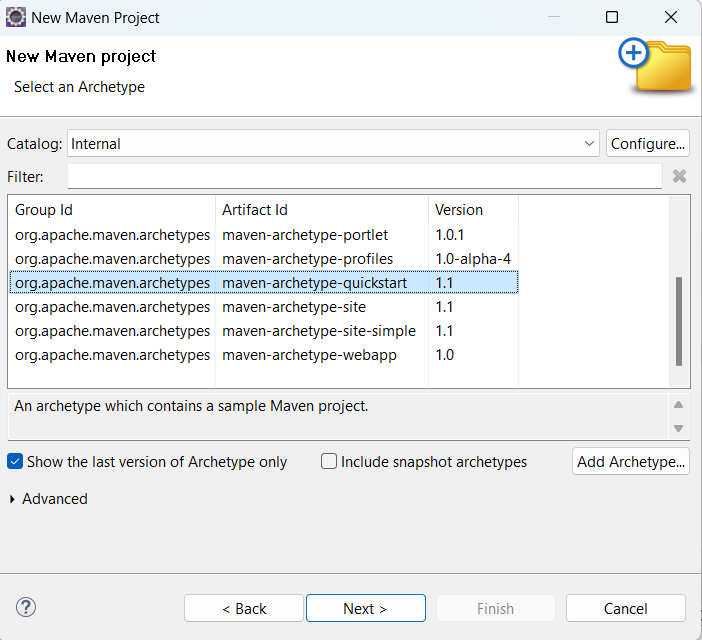
* 1. **Open Eclipse IDE, Navigate to File, then New, then Others., Select Maven Project, Click on the “Next” button.**



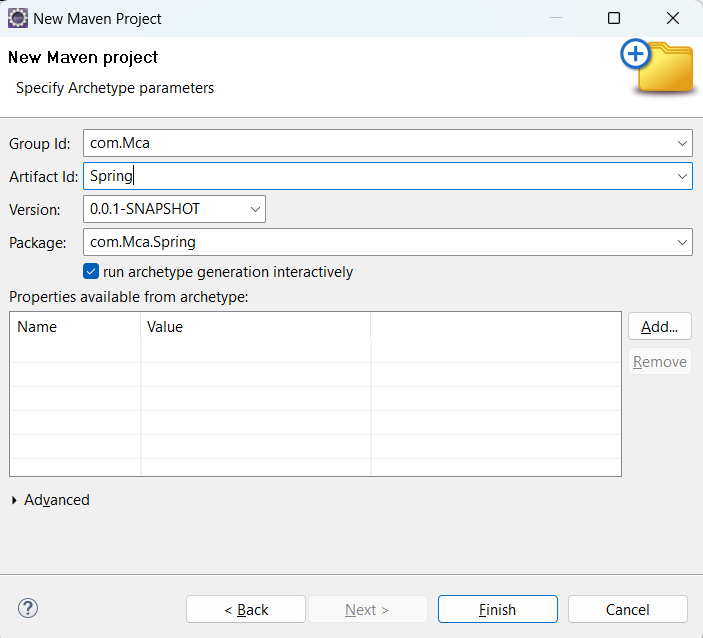
* 1. **Check the option ‘use default workspace location or choose your desired workspace location.**



* 1. **Select catalog Internal the archetype ‘maven-archetype-quickstart’.Click on the “Next” button**



* 1. **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</](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"](http://www.springframework.org/schema/beans)* xmlns:xsi=*["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance"](http://www.w3.org/2001/XMLSchema-instance)* xmlns:context=*["http://www.springframework.org/schema/context"](http://www.springframework.org/schema/context)* xmlns:p=*["http://www.springframework.org/schema/p"](http://www.springframework.org/schema/p)* xmlns:c=*["http://www.springframework.org/schema/c"](http://www.springframework.org/schema/c)* xsi:schemaLocation=*["http://www.springframework.org/schema/beans](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"](http://www.springframework.org/schema/context/spring-context.xsd)*>

<bean class=*"MCA.Singer"* name=*"singer"* p:Name=*"Sagar"* p:Age=*"23"*/>

</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**

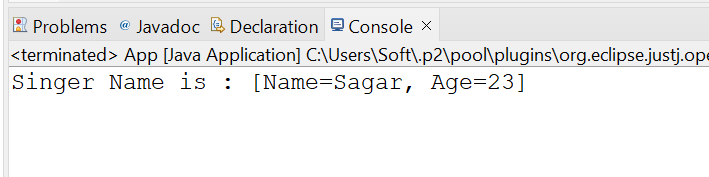
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");

Singer temp = (Singer) context.getBean("singer"); System.out.println(temp);

}

}

## Output :



### 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"](http://www.springframework.org/schema/beans)* xmlns:xsi=*["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance"](http://www.w3.org/2001/XMLSchema-instance)* xmlns:context=*["http://www.springframework.org/schema/context"](http://www.springframework.org/schema/context)* xmlns:p=*["http://www.springframework.org/schema/p"](http://www.springframework.org/schema/p)* xmlns:c=*["http://www.springframework.org/schema/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"](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) {

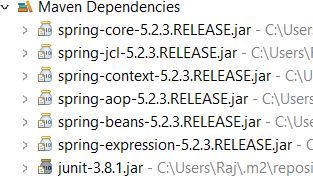
ApplicationContext context = **new**

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

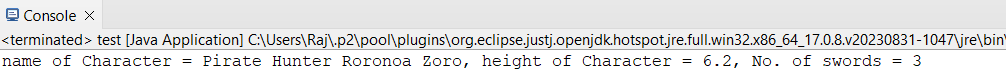
}

}

## Maven Dependencies



**Output :**



### 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"](http://www.springframework.org/schema/beans)* xmlns:xsi=*["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance"](http://www.w3.org/2001/XMLSchema-instance)* xmlns:context=*["http://www.springframework.org/schema/context"](http://www.springframework.org/schema/context)* xmlns:p=*["http://www.springframework.org/schema/p"](http://www.springframework.org/schema/p)* xmlns:c=*["http://www.springframework.org/schema/c"](http://www.springframework.org/schema/c)* xsi:schemaLocation=*["http://www.springframework.org/schema/beans](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"](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**

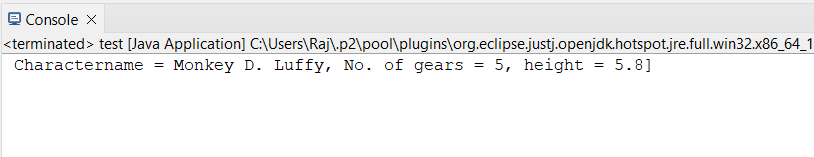
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");

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

}

}

## Output :



### 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"](http://www.springframework.org/schema/beans)* xmlns:xsi=*["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance"](http://www.w3.org/2001/XMLSchema-instance)* xmlns:context=*["http://www.springframework.org/schema/context"](http://www.springframework.org/schema/context)* xmlns:p=*["http://www.springframework.org/schema/p"](http://www.springframework.org/schema/p)* xmlns:c=*["http://www.springframework.org/schema/c"](http://www.springframework.org/schema/c)* xsi:schemaLocation=*["http://www.springframework.org/schema/beans](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"](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**

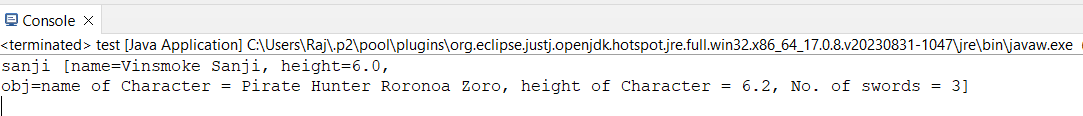
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");

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

}

}

## Output :



### 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"](http://www.springframework.org/schema/beans)* xmlns:xsi=*["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance"](http://www.w3.org/2001/XMLSchema-instance)* xmlns:context=*["http://www.springframework.org/schema/context"](http://www.springframework.org/schema/context)* xmlns:p=*["http://www.springframework.org/schema/p"](http://www.springframework.org/schema/p)* xmlns:c=*["http://www.springframework.org/schema/c"](http://www.springframework.org/schema/c)* xsi:schemaLocation=*["http://www.springframework.org/schema/beans](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"](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**

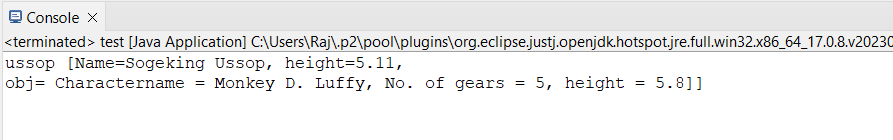
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");

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

}

}

## Output :



### 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"](http://www.springframework.org/schema/beans)* xmlns:xsi=*["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance"](http://www.w3.org/2001/XMLSchema-instance)* xmlns:context=*["http://www.springframework.org/schema/context"](http://www.springframework.org/schema/context)* xmlns:p=*["http://www.springframework.org/schema/p"](http://www.springframework.org/schema/p)* xmlns:c=*["http://www.springframework.org/schema/c"](http://www.springframework.org/schema/c)* xsi:schemaLocation=*["http://www.springframework.org/schema/beans](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"](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**

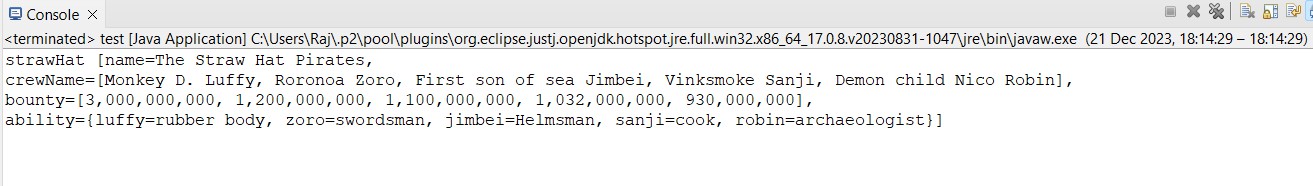
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");

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

}

}

## Output :



### 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"](http://www.springframework.org/schema/beans)* xmlns:xsi=*["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance"](http://www.w3.org/2001/XMLSchema-instance)* xmlns:context=*["http://www.springframework.org/schema/context"](http://www.springframework.org/schema/context)* xmlns:p=*["http://www.springframework.org/schema/p"](http://www.springframework.org/schema/p)* xmlns:c=*["http://www.springframework.org/schema/c"](http://www.springframework.org/schema/c)* xsi:schemaLocation=*["http://www.springframework.org/schema/beans](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"](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**

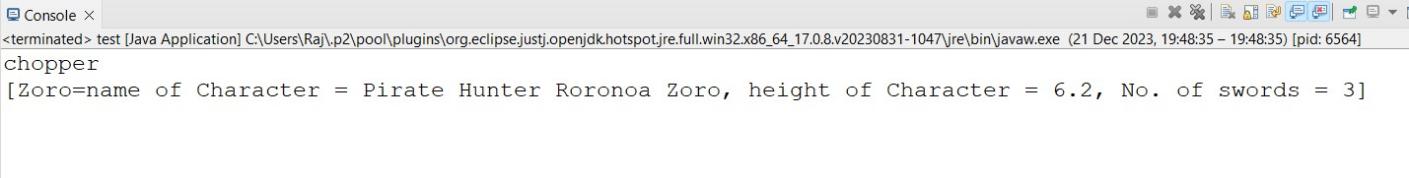
ClassPathXmlApplicationContext("MCA/mcaConfig.xml");

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

}

}

## Output :

x

# PRACTICAL 8: ASSIGNMENT BASED ASPECT ORIENTED PROGRAMMING

### 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>

<!-- 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"](http://www.springframework.org/schema/beans)* xmlns:xsi=*["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance"](http://www.w3.org/2001/XMLSchema-instance)* xmlns:context=*["http://www.springframework.org/schema/context"](http://www.springframework.org/schema/context)* xmlns:aop=*["http://www.springframework.org/schema/aop"](http://www.springframework.org/schema/aop)*

xsi:schemaLocation=*["http://www.springframework.org/schema/beans](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**

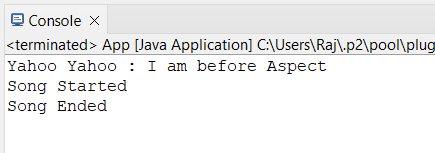
ClassPathXmlApplicationContext("aop/aopConfig.xml");

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

}

}

## Output :



### 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())")

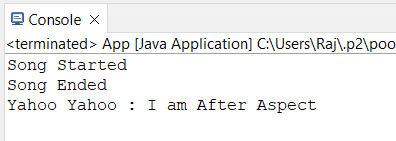
**public void** afterSong() {

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

}

}

## Output :



### 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())")

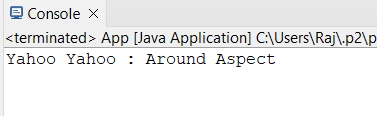
**public void** aroundSong() {

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

}

}

## Output :



### 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())")

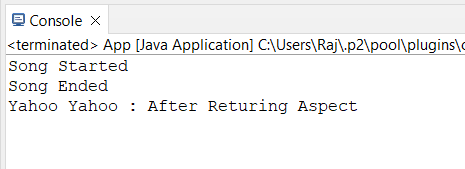
**public void** AfterReturnSong() {

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

}

}

## Output :



### 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 :



### 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()")

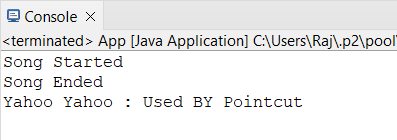
**public void** afterSong() {

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

}

}

**Output :**



# PRACTICAL 9 : ASSIGNMENT BASED SPRING JDBC

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

## Pom.xml

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

xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> <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</](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"](http://www.springframework.org/schema/beans)* xmlns:xsi=*["http://www.w3.org/2001/XMLSchema](http://www.w3.org/2001/XMLSchema-instance)-[instance"](http://www.w3.org/2001/XMLSchema-instance)* xmlns:context=*["http://www.springframework.org/schema/context"](http://www.springframework.org/schema/context)* xmlns:p=*["http://www.springframework.org/schema/p"](http://www.springframework.org/schema/p)* xmlns:c=*["http://www.springframework.org/schema/c"](http://www.springframework.org/schema/c)* xsi:schemaLocation=*["http://www.springframework.org/schema/beans](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"](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);

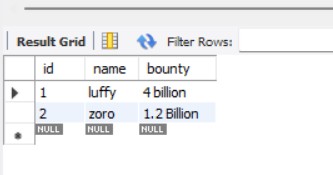
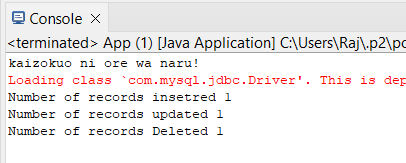
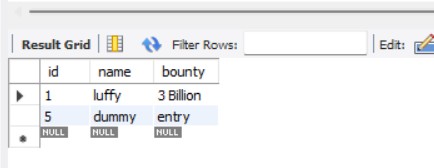
**int** result2 = temp.update(query2,"4 billion",1); System.***out***.println("Number of records updated " + result2);

**int** result3 = temp.update(query3,5); System.***out***.println("Number of records Deleted " + result3);

}

}

## Output:



### 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");

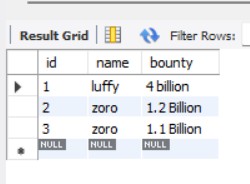
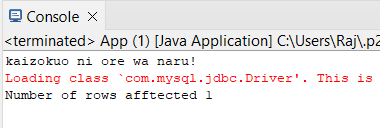
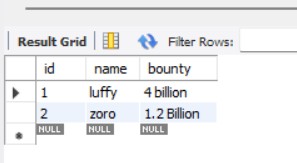
**return** ps;

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

}

}

**Output:**

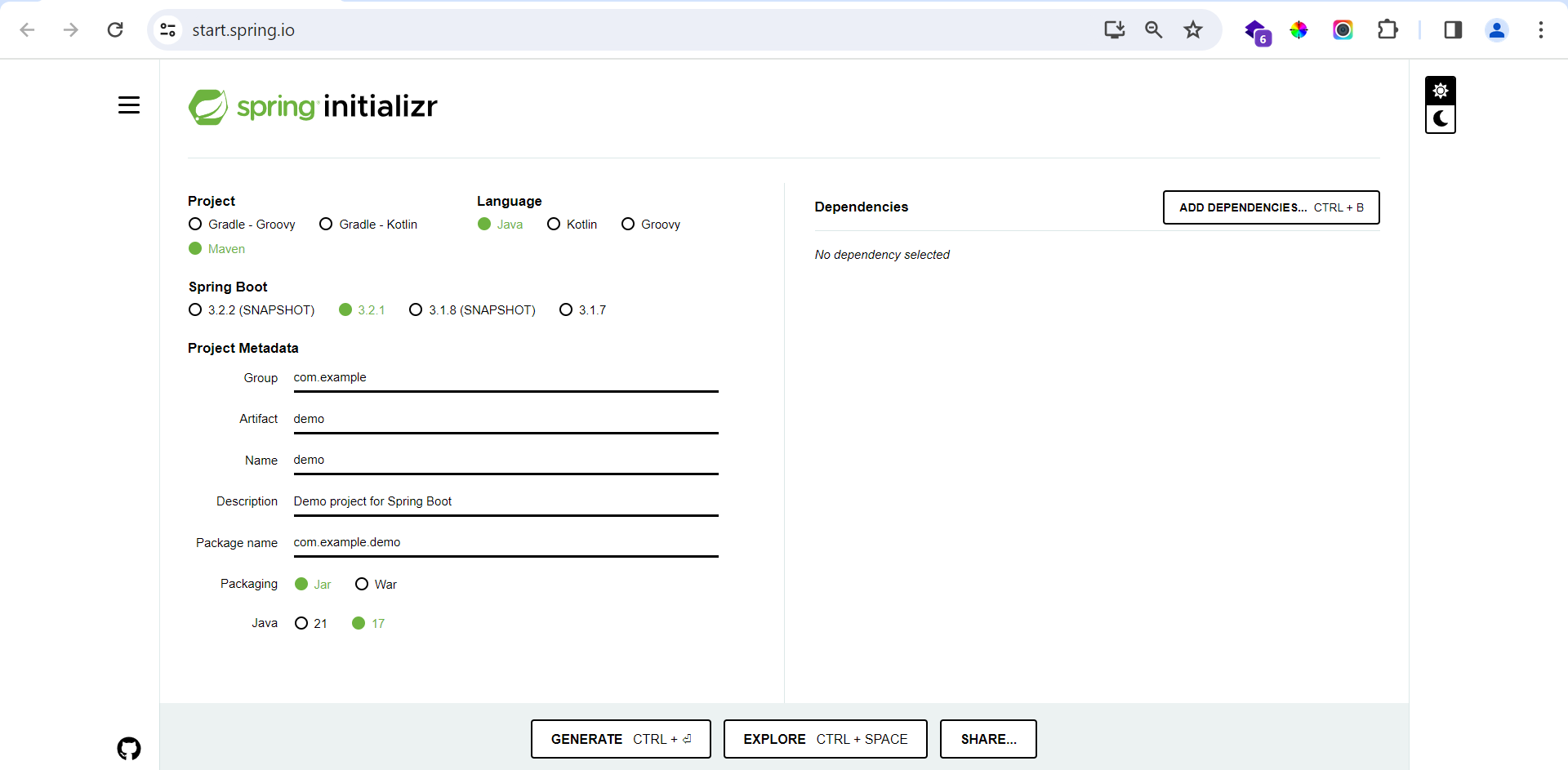


# PRACTICAL 10: ASSIGNMENT BASED SPRING BOOT AND RESTFUL WEB SERVICES

### Write a program to create a simple Spring Boot application prints a message

1. **Go to [Spring Initializr](https://start.spring.io/). Select the type of project (Maven). Choose the language (Java). Select the Spring Boot version.**

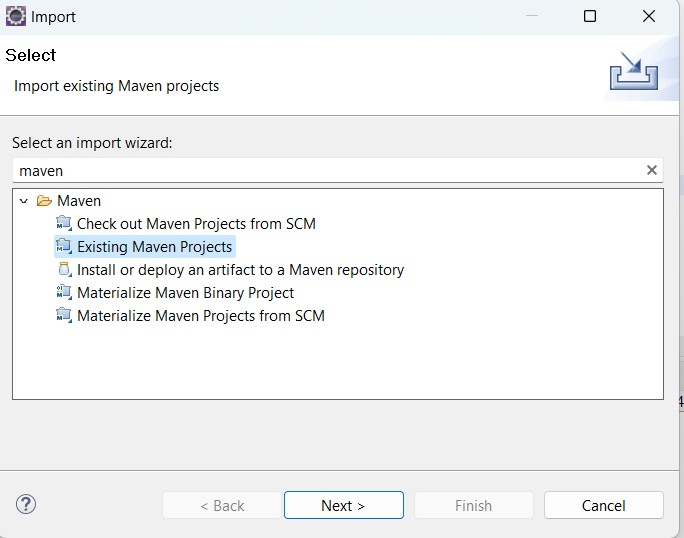
**Fill in the project metadata. Add the necessary dependencies (at least spring-boot- starter-web). Click on “Generate” to download the project.**

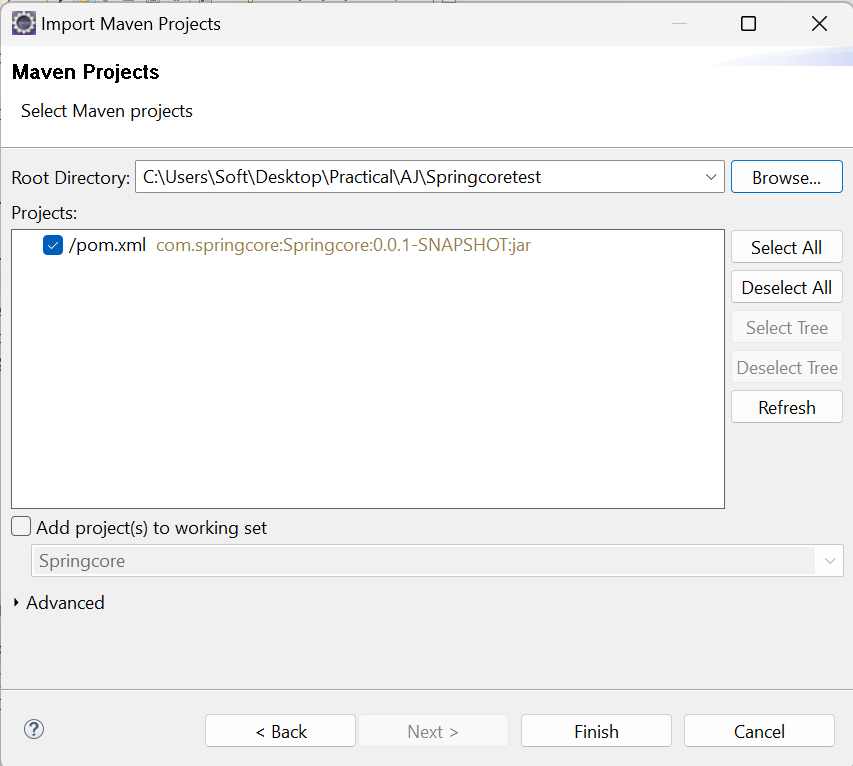


1. **Open Eclipse IDE. Navigate to File > Import.**

**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 :**

