

# **Advanced Java Lab**

## **Subject Code: MCAL12**

A Practical Journal Submitted in Fulfilment of  
the Degree of

**MASTER**

**In**

**COMPUTER APPLICATION**

Year 2023-2024

By

**Vishwakarma Dheeraj Kumar Jaynath**

**(78383)**

Semester- 1

Under the Guidance of

**Prof.**



Institute of Distance and Open Learning

Vidya Nagari, Kalina, Santacruz East – 400098.

University of Mumbai

**PCP Centre**

[Vidyavardhini's College of Technology – Vasai Road, Palghar 401202]



**Institute of Distance and Open Learning,**  
**Vidya Nagari, Kalina, Santacruz (E) -400098**

**CERTIFICATE**

This to certify that, **Vishwakarma Dheeraj Kumar Jaynath** appearing  
**Master in Computer Application (Semester I) 78383** has satisfactory completed  
the prescribed practical of **MCAL12-Advanced Java Lab** as laid down by the  
University of Mumbai for the academic year 2023-24

Teacher in charge

Examiners

Coordinator  
IDOL, MCA  
University of Mumbai

Date:

Place:

# Index

Sr. No.	Practical	Signature
1.	Write a Java Program to demonstrate a Generic Class and Generic Methods.	
2.	Write a Java program to create List containing list of items and use List Iterator interface to print items present in the list. Also print the list in reverse/backward direction.	
3.	Write a Java program using Lambda Expression to calculate the following: a. Convert Fahrenheit to Celsius b. Convert Kilometers to Miles.	
4.	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	
5.	Write a JSP page to display the Registration form (Make your own assumptions).	
6.	Create a EJB class to accept name, address and email of a user and to display it.	
7.	Write a program to create Java POJO class.	
8.	Write a program to print "Hello World" using spring framework.	
9.	Write a program to insert, update and delete records from the given table.	
10.	Write a program to demonstrate Spring AOP before advice.	

## Practical No: 1

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

**Objective:** To create a Java Program that illustrates the use of Generic Classes and Generic Methods.

**Description:** we will implement a generic class and generic methods to demonstrate the power of generics in Java. The generic class will be designed to handle various types of data, ensuring type safety at compile-time. Additionally, generic methods within the program will showcase how methods can be written to operate on a variety of data types without sacrificing code clarity or safety. Through this demonstration, we aim to emphasize the benefits of code reusability and flexibility provided by generics in Java.

### Program:

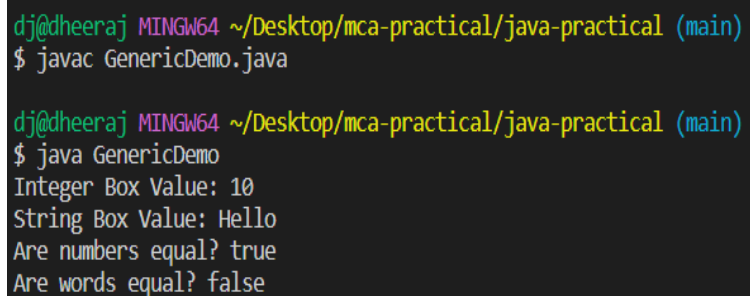
```
public class GenericDemo {
    static class GenericBox<T> {
        private T value;
        public GenericBox(T value) {
            this.value = value;
        }
        public T getValue() {
            return value;
        }
    }

    static <T> boolean areEqual(T element1, T element2) {
        return element1.equals(element2);
    }
}
```

```
public static void main(String[] args) {
    GenericBox<Integer> integerBox = new GenericBox<>(10);
    GenericBox<String> stringBox = new GenericBox<>("Hello");

    System.out.println("Integer Box Value: " + integerBox.getValue());
    System.out.println("String Box Value: " + stringBox.getValue());
    Integer number1 = 5;
    Integer number2 = 5;
    String word1 = "Hello";
    String word2 = "World";

    System.out.println("Are numbers equal? " + areEqual(number1, number2));
    System.out.println("Are words equal? " + areEqual(word1, word2));
}
}
```



```
dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ javac GenericDemo.java

dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ java GenericDemo
Integer Box Value: 10
String Box Value: Hello
Are numbers equal? true
Are words equal? false
```

### Conclusion:

## Practical No: 2

**Aim:** 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.

**Objective:** Develop a Java program to create a List containing items and utilize the ListIterator interface to print the items both in the forward and reverse directions.

**Description:** This Java program involves the creation of a List, populating it with items, and utilizing the ListIterator interface to traverse and print the items in both forward and backward directions. The ListIterator facilitates sequential access to the elements in the List, enabling efficient navigation and printing.

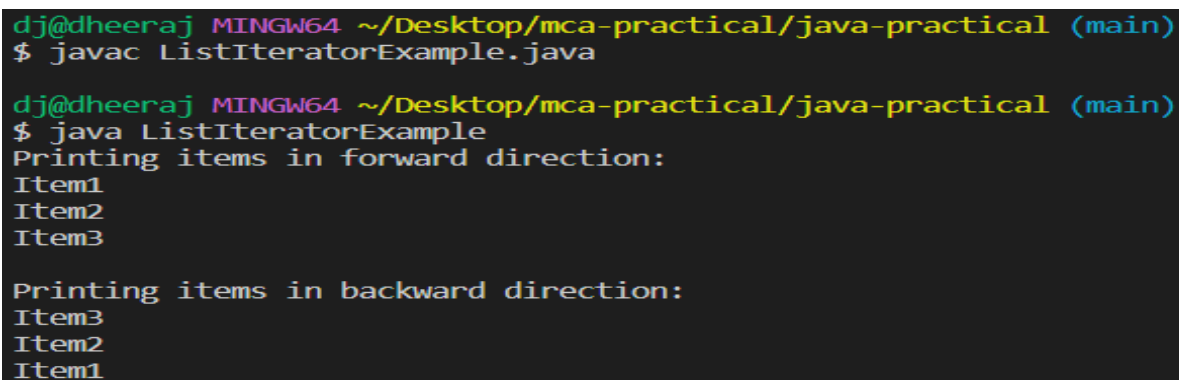
### Program:

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class ListIteratorExample {
    public static void main(String[] args) {
        List<String> itemList = new ArrayList<>();
        itemList.add("Item1");
        itemList.add("Item2");
        itemList.add("Item3");

        System.out.println("Printing items in forward direction:");
        ListIterator<String> forwardIterator = itemList.listIterator();
        while (forwardIterator.hasNext()) {
            System.out.println(forwardIterator.next());
        }

        System.out.println("\nPrinting items in backward direction:");
        ListIterator<String> backwardIterator = itemList.listIterator(itemList.size());
        while (backwardIterator.hasPrevious()) {
            System.out.println(backwardIterator.previous());
        }
    }
}
```



```
dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ javac ListIteratorExample.java

dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ java ListIteratorExample
Printing items in forward direction:
Item1
Item2
Item3

Printing items in backward direction:
Item3
Item2
Item1
```

### Conclusion:

### Practical No: 3

**Aim:** Write a Java program using Lambda Expression to calculate the following: a. Convert Fahrenheit to Celsius b. Convert Kilometers to Miles.

**Objective:** Create a Java program using Lambda Expressions to perform temperature conversion from Fahrenheit to Celsius and distance conversion from kilometers to miles.

**Description:** This Java program utilizes Lambda Expressions to implement functions for converting Fahrenheit to Celsius and kilometers to miles. The concise syntax of Lambda Expressions enhances code readability and conciseness, making it an efficient approach for such calculations.

#### Program:

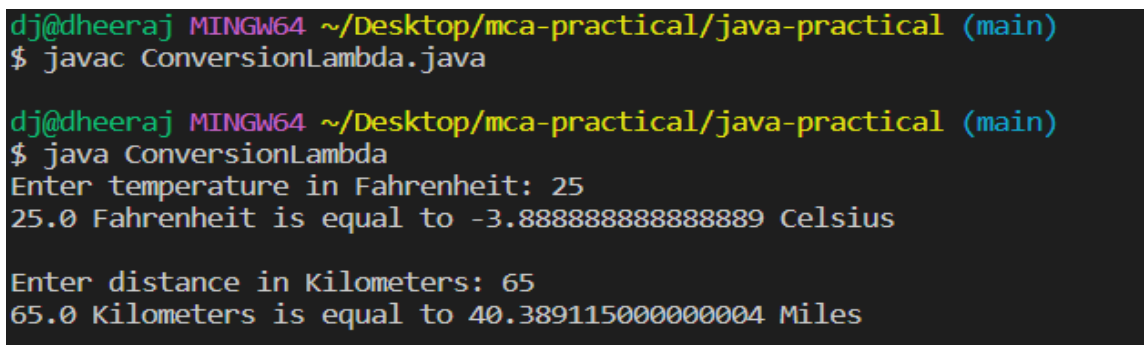
```
import java.util.Scanner;
interface Converter {
    double convert(double input);
}

public class ConversionLambda {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Converter fahrenheitToCelsius = (fahrenheit) -> (fahrenheit - 32.0) * 5.0 / 9.0;
        Converter kilometersToMiles = (kilometers) -> kilometers * 0.621371;

        System.out.print("Enter temperature in Fahrenheit: ");
        double fahrenheit = scanner.nextDouble();
        double celsius = fahrenheitToCelsius.convert(fahrenheit);
        System.out.println(fahrenheit + " Fahrenheit is equal to " + celsius + " Celsius");

        System.out.print("\nEnter distance in Kilometers: ");
        double kilometers = scanner.nextDouble();
        double miles = kilometersToMiles.convert(kilometers);
        System.out.println(kilometers + " Kilometers is equal to " + miles + " Miles");

        scanner.close();
    }
}
```



```
dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ javac ConversionLambda.java

dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ java ConversionLambda
Enter temperature in Fahrenheit: 25
25.0 Fahrenheit is equal to -3.888888888888889 Celsius

Enter distance in Kilometers: 65
65.0 Kilometers is equal to 40.389115000000004 Miles
```

#### Conclusion:

## Practical No: 4

**Aim:** 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.

**Objective:** Create a Java program utilizing the Map interface to manage a collection of items with keys & associated values, performing operations to add items to the map and remove items from it.

**Description:** This Java program leverages the Map interface to handle a collection of items where each item is associated with a unique key. The program includes functionality to add items to the map and remove items from it, demonstrating the versatility of the Map interface in managing key-value pairs.

### Program:

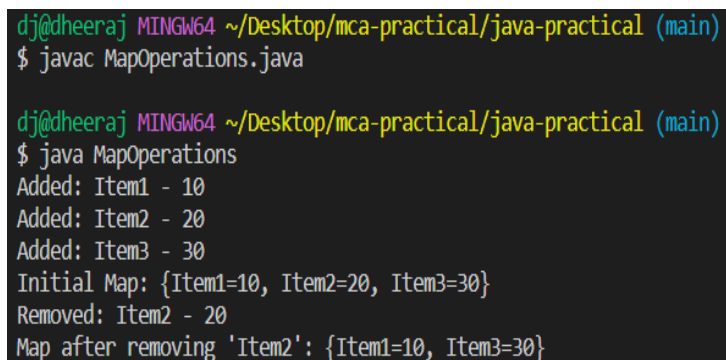
```
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
```

```
public class MapOperations {
    public static void main(String[] args) {
        Map<String, Integer> itemMap = new HashMap<>();

        addItemToMap(itemMap, "Item1", 10);
        addItemToMap(itemMap, "Item2", 20);
        addItemToMap(itemMap, "Item3", 30);
        System.out.println("Initial Map: " + itemMap);
        removeItemFromMap(itemMap, "Item2");
        System.out.println("Map after removing 'Item2': " + itemMap);
    }

    private static void addItemToMap(Map<String, Integer> map, String key, int value) {
        map.put(key, value);
        System.out.println("Added: " + key + " - " + value);
    }

    private static void removeItemFromMap(Map<String, Integer> map, String key) {
        if (map.containsKey(key)) {
            int removedValue = map.remove(key);
            System.out.println("Removed: " + key + " - " + removedValue);
        } else {
            System.out.println("Item '" + key + "' not found in the map.");
        }
    }
}
```



```
dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ javac MapOperations.java

dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ java MapOperations
Added: Item1 - 10
Added: Item2 - 20
Added: Item3 - 30
Initial Map: {Item1=10, Item2=20, Item3=30}
Removed: Item2 - 20
Map after removing 'Item2': {Item1=10, Item3=30}
```

### Conclusion:

## Practical No: 5

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

**Objective:** Develop a JSP (Java Server Pages) page to display a registration form.

**Description:** This JSP page is designed to present a user-friendly registration form, incorporating HTML and JSP tags to create an interactive and visually appealing interface. The form may include fields for user details such as name, email, password, and other relevant information, allowing users to submit their registration information.

**Program:**

```
WebApplication1 - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Start Page * index.html *
Source History

8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta charset="UTF-8">
12 <title>Registration Form</title>
13 </head>
14 <body>
15 <h2>Registration Form</h2>
16 <form action="registration_process.jsp" method="post">
17 <label for="firstName">First Name:</label>
18 <input type="text" id="firstName" name="firstName" required><br>
19
20 <label for="lastName">Last Name:</label>
21 <input type="text" id="lastName" name="lastName" required><br>
22
23 <label for="email">Email:</label>
24 <input type="email" id="email" name="email" required><br>
25
26 <label for="password">Password:</label>
27 <input type="password" id="password" name="password" required><br>
28
29 <label for="confirmPassword">Confirm Password:</label>
30 <input type="password" id="confirmPassword" name="confirmPassword" required><br>
31
32 <label for="gender">Gender:</label>
33 <select id="gender" name="gender">
34 <option value="male">Male</option>
35 <option value="female">Female</option>
36 <option value="other">Other</option>
37 </select><br>
38
39 <label for="birthdate">Date of Birth:</label>
40 <input type="date" id="birthdate" name="birthdate" required><br>
41
42 <input type="submit" value="Register">
43 </form>
44 </body>
45 </html>
46
```

## Registration Form

First Name:

Last Name:

Email:

Password:

Confirm Password:

Gender:  ▼

Date of Birth:

**Conclusion:**



## Practical No: 6

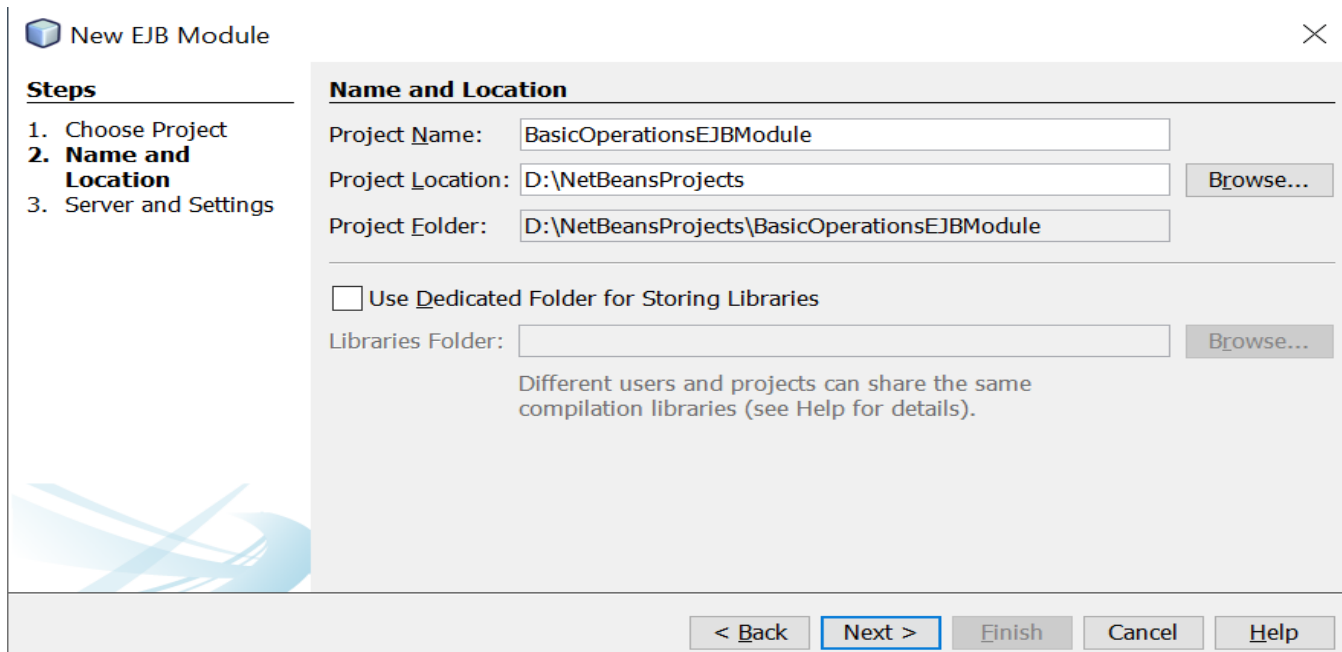
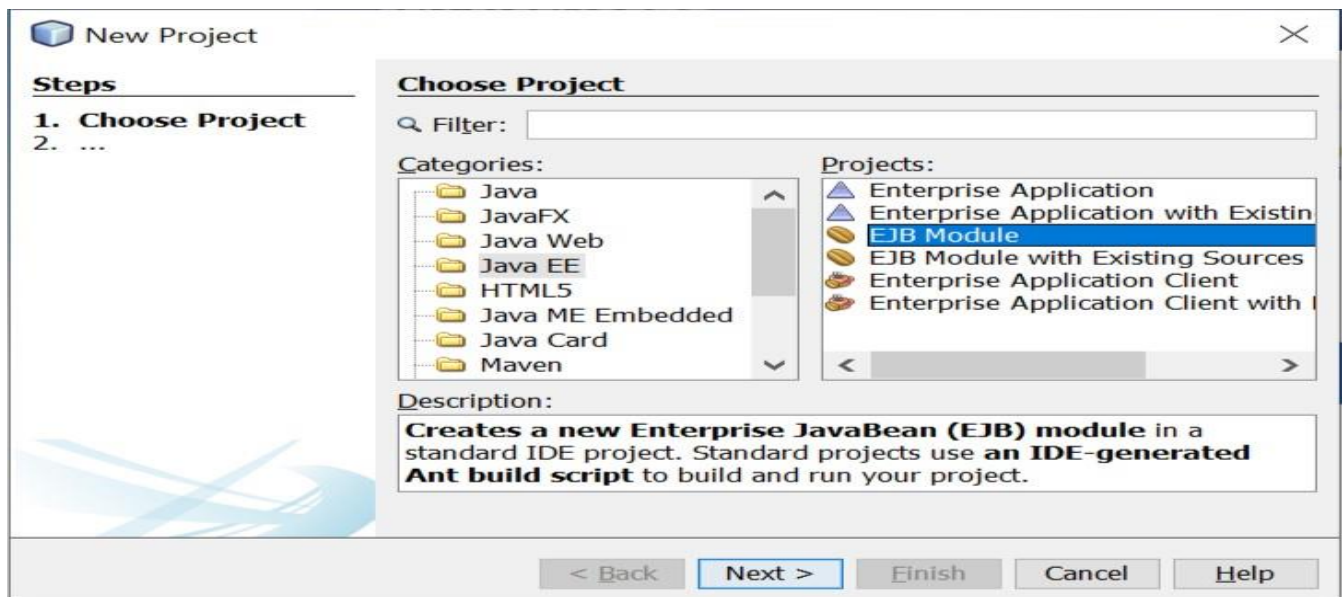
**Aim:** Create a EJB class to accept name, address and email of a user and to display it.

**Objective:** Develop an Enterprise JavaBeans (EJB) class to receive and store user information, including name, address, and email, and implement a method to display the stored information.

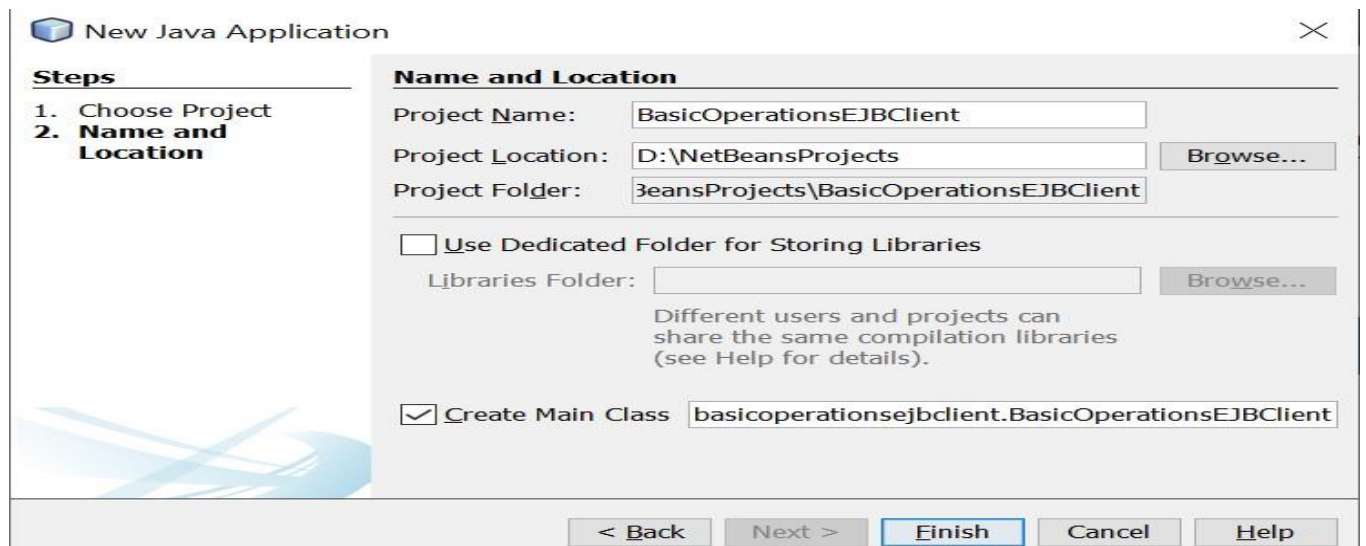
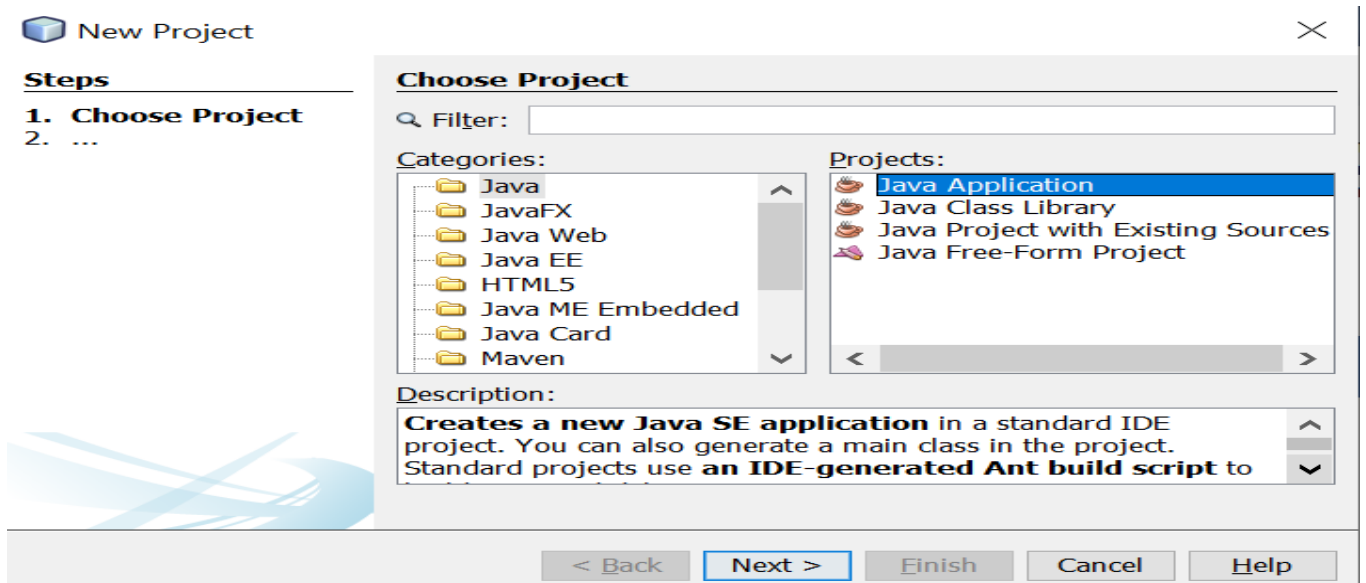
**Description:** This EJB class is designed to encapsulate user information by accepting inputs for name, address, and email. It includes methods to store this information and retrieve it for display. The EJB architecture facilitates the creation of scalable and reusable components for managing user data within enterprise applications.

### Program:

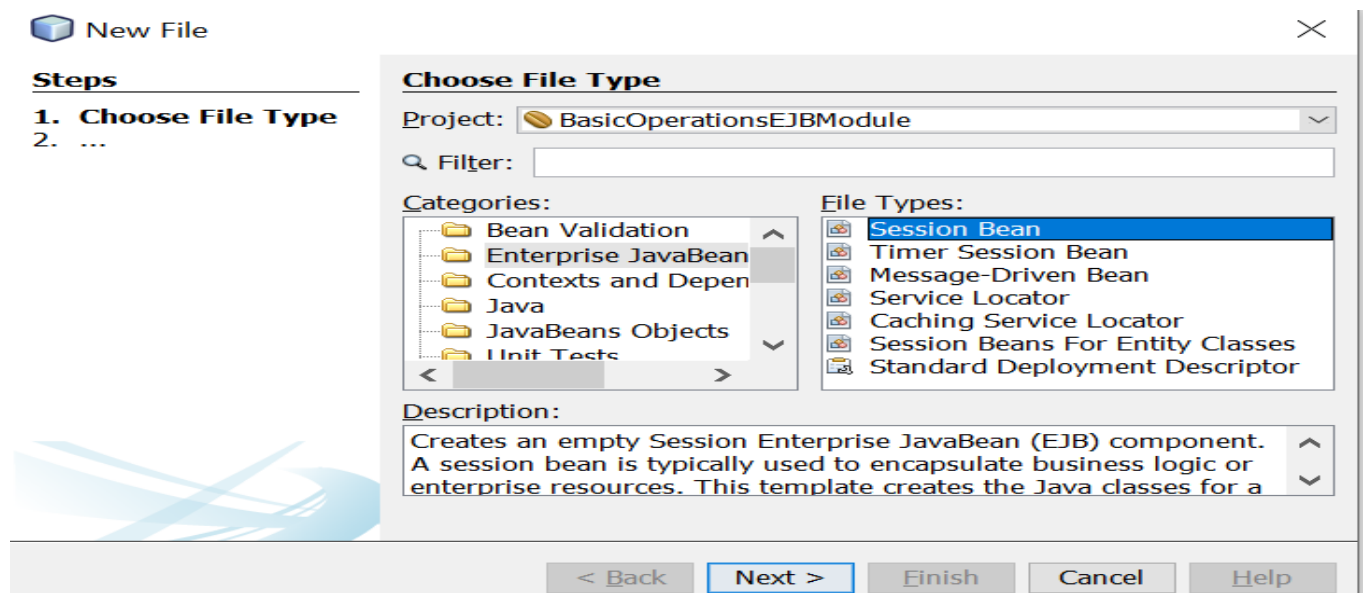
1. Create EJB Module.



## 2. Create Java Application.



## 3. Create Session Bean.



**New Session Bean**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

EJB Name:

Project:

Location:

Package:

Session Type:

☒ Stateless

☐ Stateful

☐ Singleton

Create Interface:

☐ Local

☐ Remote

< Back Next > Finish Cancel Help

#### 4. Add Methods.

BasicOperationsEJBModule - NetBeans IDE 8.0.2

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Start Page BasicOperationsEJBClient.java OperationsSessionBean.java

Source History

```

1  *
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package com.javacodegeeks.example.ejb;
7
8  import javax.ejb.Stateless;
9
10 /**
11  *
12  * @author SWAGAT
13  */
14 @Stateless
15 public class OperationsSessionBean implements OperationsSessionBeanRemote {
16
17     @Override
18     public float add(float x, float y) {
19         return x + y;
20     }
21
22     @Override
23     public float subtract(float x, float y) {
24         return x - y;
25     }
26
27     @Override
28     public float multiply(float x, float y) {
29         return x * y;
30     }
31
32     @Override
33     public float divide(float x, float y) {
34         return x / y;
35     }
36
37 }

```

#### 5. Deploy EJB Module.

#### 6. Create Web Application.

**New Web Application**

**Steps**

1. Choose Project
2. **Name and Location**
3. Server and Settings
4. Frameworks

**Name and Location**

Project Name:

Project Location:  Browse...

Project Folder:

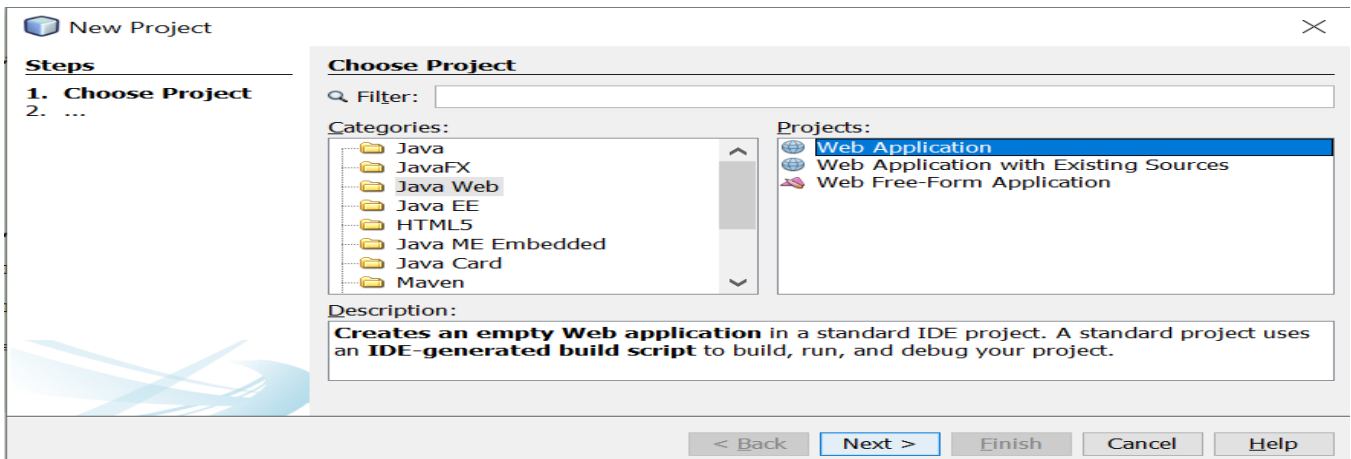
☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:  Browse...

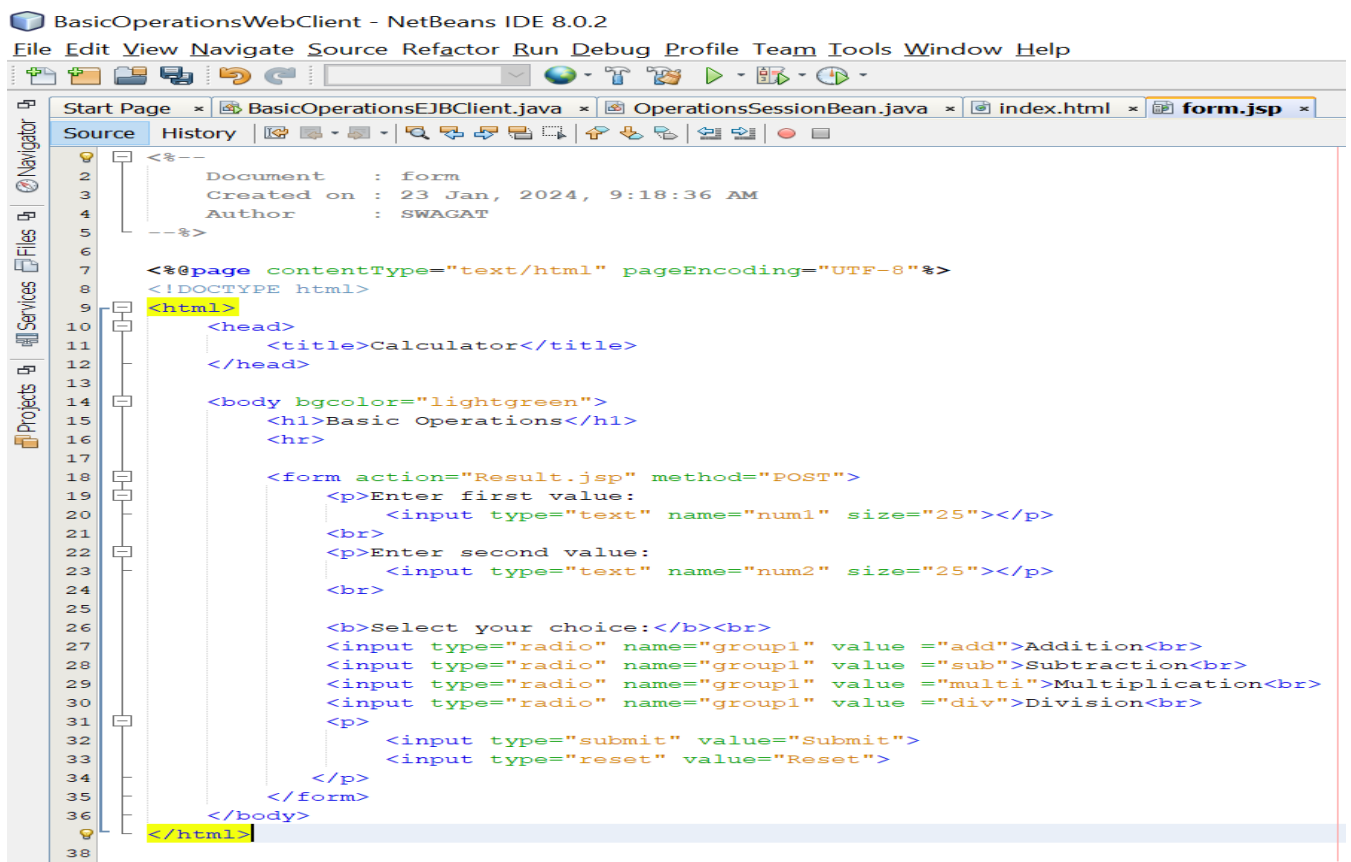
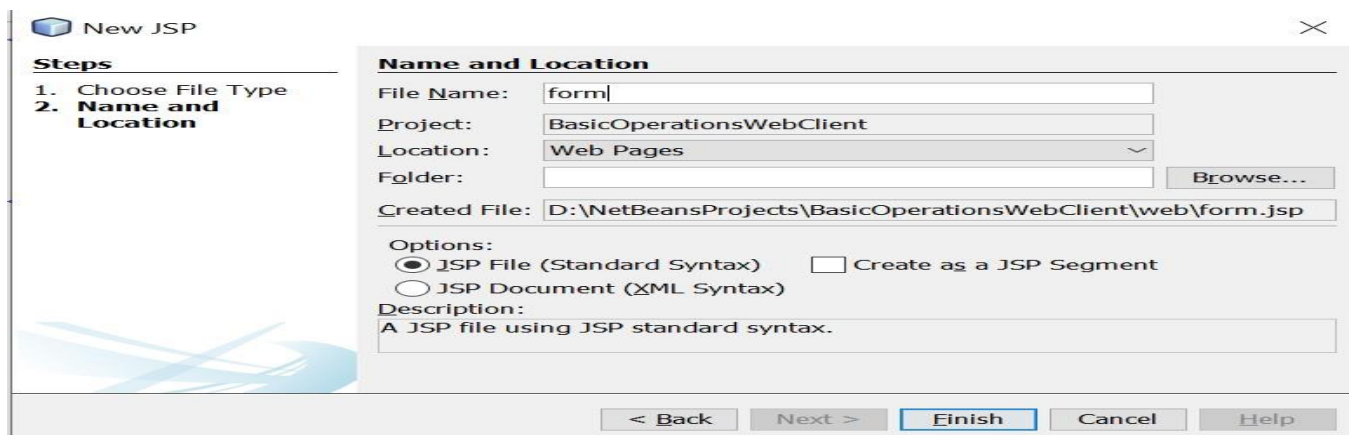
Different users and projects can share the same compilation libraries (see Help for details).

Project Folder already exists and is not empty.

< Back Next > Finish Cancel Help



## 7. Create JSP Files.



## Steps

1. Choose File Type
2. Name and Location

## Name and Location

File Name:

Project:

Location:  Browse...

Folder:

Created File:

Options:

☒ JSP File (Standard Syntax) ☐ Create as a JSP Segment

☐ JSP Document (XML Syntax)

Description:

A JSP file using JSP standard syntax.

&lt; Back

Next &gt;

Finish

Cancel

Help

Start Page x BasicOperationsEJBClient.java x OperationsSessionBean.java x index.html x form.jsp x Result.jsp x

Source History

```

7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <%@ page import="com.javacodegeeks.example.ejb.*, javax.naming.*"%>
9
10 <%=
11     private OperationsSessionBeanRemote ops = null;
12     float result = 0;
13
14     public void jspInit() {
15         try {
16
17             InitialContext ic = new InitialContext();
18             ops = (OperationsSessionBeanRemote)ic.lookup(OperationsSessionBeanRemote.class.getName());
19
20             System.out.println("Loaded Calculator Bean");
21
22
23         } catch (Exception ex) {
24             System.out.println("Error:"
25                 + ex.getMessage());
26         }
27     }
28
29     public void jspDestroy() {
30         ops = null;
31     }
32 }
33 %>
34
35
36 <%
37
38     try {
39         String s1 = request.getParameter("num1");
40         String s2 = request.getParameter("num2");
41         String s3 = request.getParameter("group1");
42
43         System.out.println(s3);
44
45         if (s1 != null && s2 != null) {
46             Float num1 = new Float(s1);

```

Breakpoints Variables

Start Page x BasicOperationsEJBClient.java x OperationsSessionBean.java x index.html x form.jsp x

Source History

```

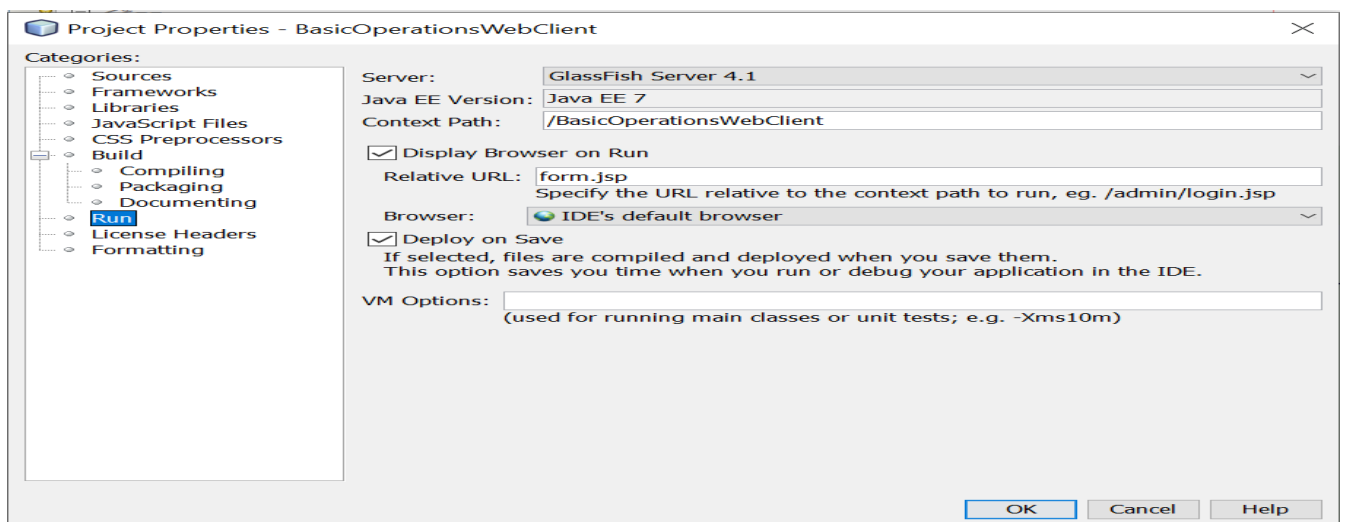
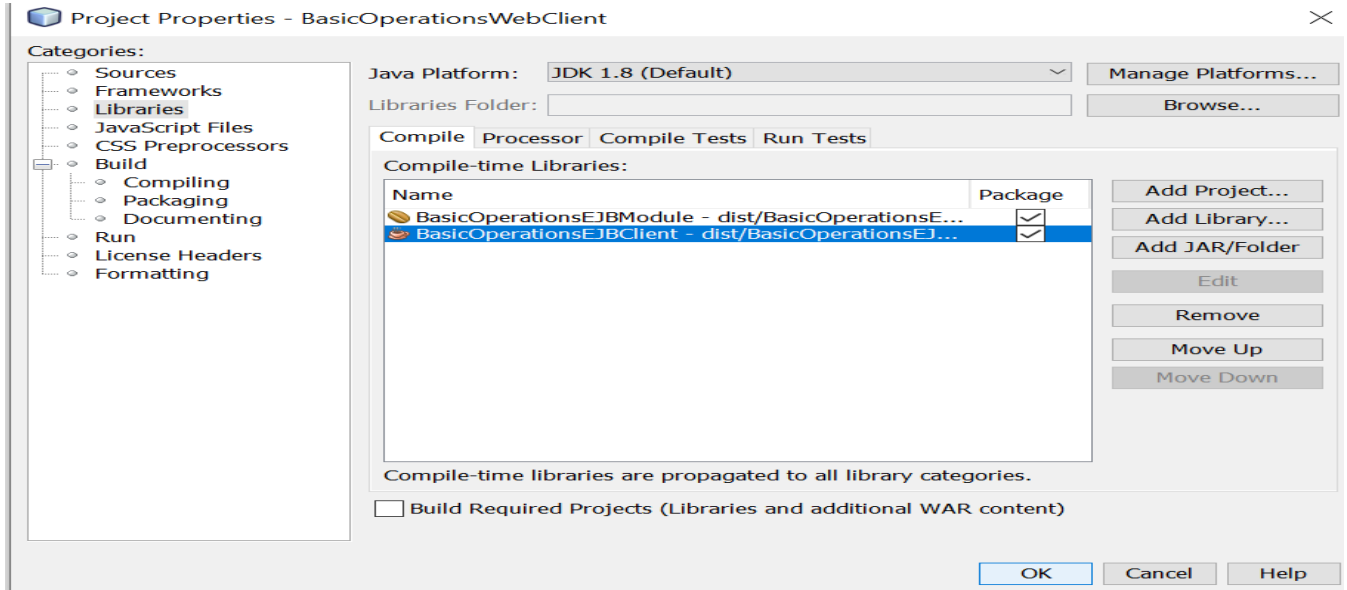
34
35
36 <%
37
38     try {
39         String s1 = request.getParameter("num1");
40         String s2 = request.getParameter("num2");
41         String s3 = request.getParameter("group1");
42
43         System.out.println(s3);
44
45         if (s1 != null && s2 != null) {
46             Float num1 = new Float(s1);
47             Float num2 = new Float(s2);
48
49             if (s3.equals("add")) {
50                 result = ops.add(num1.floatValue(), num2.floatValue());
51             } else if (s3.equals("sub")) {
52                 result = ops.subtract(num1.floatValue(), num2.floatValue());
53             } else if (s3.equals("multi")) {
54                 result = ops.multiply(num1.floatValue(), num2.floatValue());
55             } else {
56                 result = ops.divide(num1.floatValue(), num2.floatValue());
57             }
58         }
59     }
60 }
61 <p> <b>The result is:</b> <%= result%>
62 <p>
63
64 <%
65     }
66     } // end of try
67     catch (Exception e) {
68         e.printStackTrace();
69         //result = "Not valid";
70     }
71 }
72 %>

```

Breakpoints Variables



## 1. Add Libraries and Run Project.



The result is: 4.0

Conclusion:

## Practical No:7

**Aim:** Write a program to create Java POJO class.

**Objective:** Develop a Java program to create a Plain Old Java Object (POJO) class.

**Description:** This Java program focuses on creating a Plain Old Java Object (POJO) class, typically used to encapsulate data and provide getters and setters for accessing and modifying the object's attributes. A POJO class does not contain any business logic or dependencies on specific frameworks, emphasizing simplicity and reusability.

**Program:**

```
public class Person {
    private String name;
    private int age;
    private String email;

    public Person() {
        // Default constructor
    }

    public Person(String name, int age, String email) {
        this.name = name;
        this.age = age;
        this.email = email;
    }

    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

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

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

```

@Override
public String toString() {
    return "Person{" +
        "name='" + name + "\" + ", age=" + age +
        ", email='" + email + "\" + '}";
}

public static void main(String[] args) {
    Person person1 = new Person("John Doe", 25, "john.doe@example.com");
    Person person2 = new Person();

    person2.setName("Jane Doe");
    person2.setAge(30);
    person2.setEmail("jane.doe@example.com");

    System.out.println(person1);
    System.out.println(person2);
}
}

```

```

dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ javac Person.java

dj@dheeraj MINGW64 ~/Desktop/mca-practical/java-practical (main)
$ java Person
Person{name='John Doe', age=25, email='john.doe@example.com'}
Person{name='Jane Doe', age=30, email='jane.doe@example.com'}

```

**Conclusion:**



## Practical No: 8

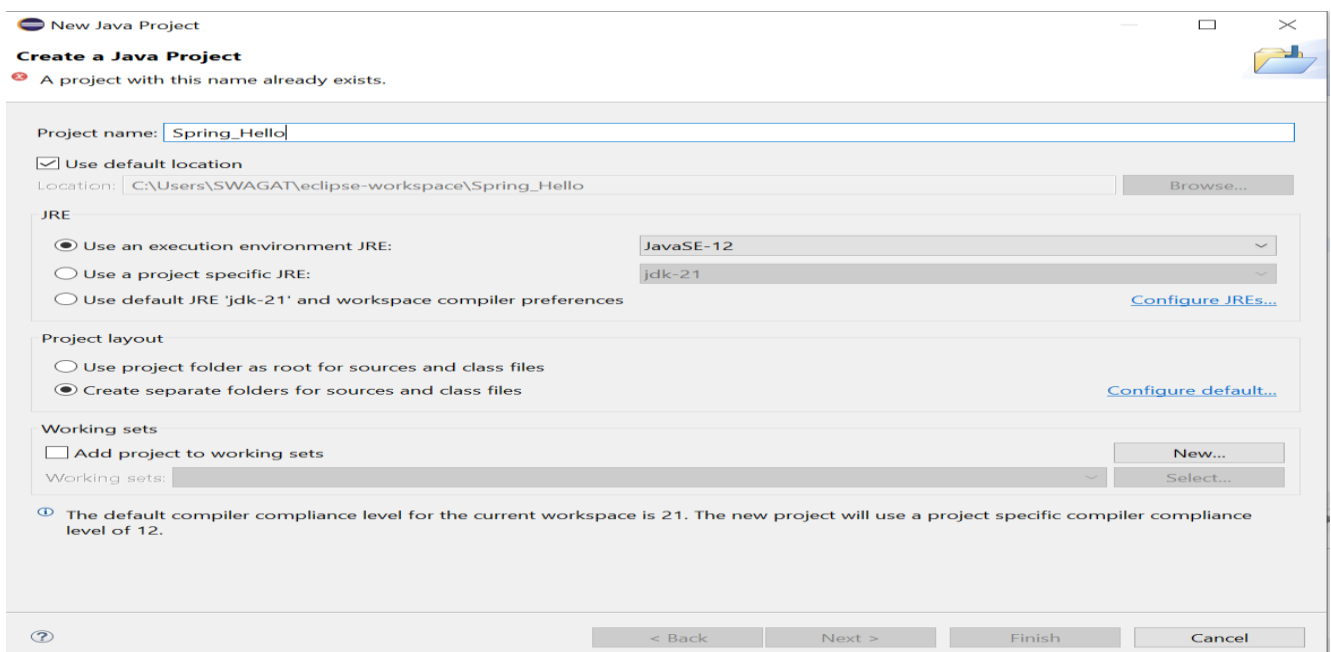
**Aim:** Write a program to print “Hello World” using spring framework.

**Objective:** Develop a Java program using the Spring framework to print the message "Hello World."

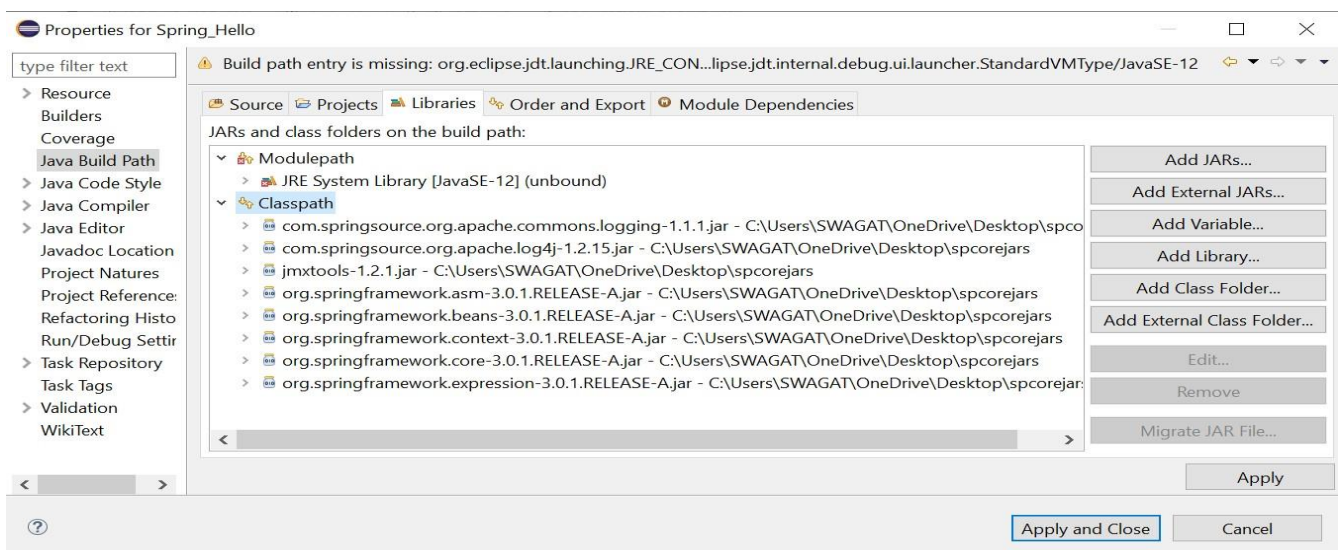
**Description:** This program utilizes the Spring framework, a widely-used framework for building Java applications, to create a simple application that prints the classic "Hello World" message. The simplicity and clarity of the Spring framework make it an effective choice for developing Java applications.

**Program:**

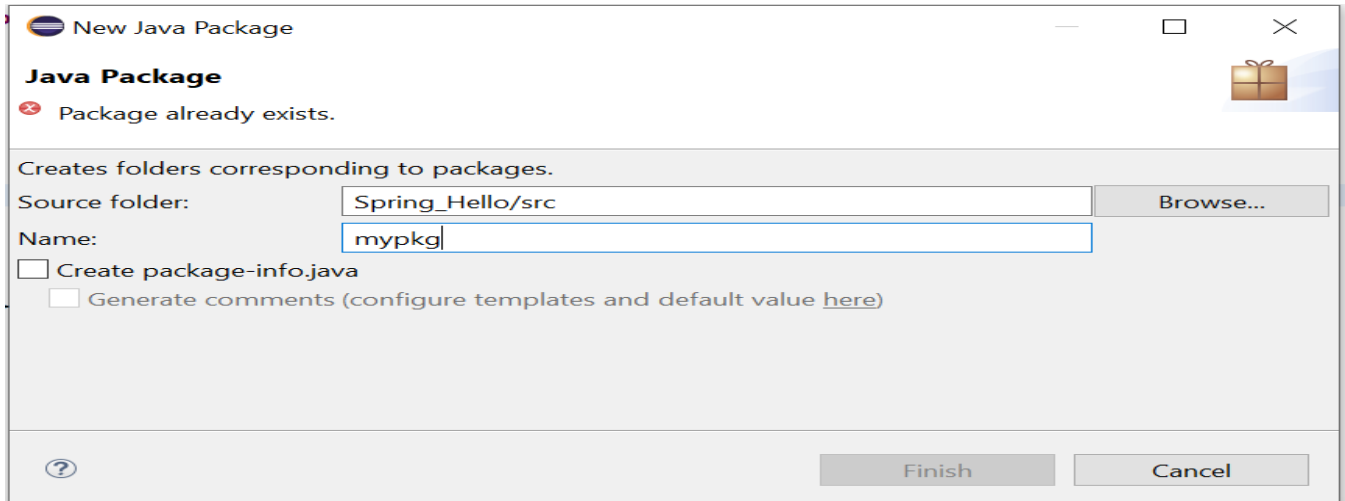
### 1. Create Java Project



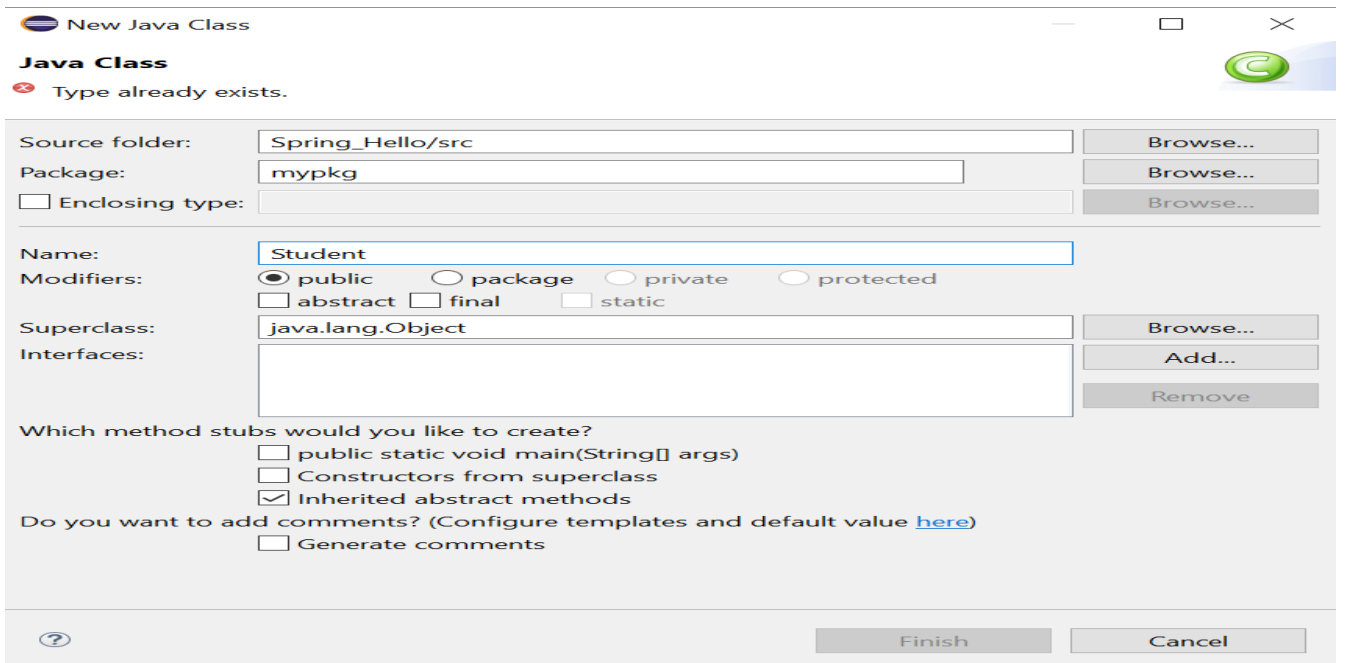
### 2. Import JAR Files.



### 3. Create package.



### 4. Create Class.



```
package mypkg;  
public class Student {
```

```
    private String name;  
    public String getName() {return name;}  
    public void setName(String name) { this.name = name;}  
    public void displayInfo() { System.out.println("Hello: " + name);}  
}
```

```
package mypkg;  
import org.springframework.beans.factory.BeanFactory;  
import org.springframework.beans.factory.xml.XmlBeanFactory;  
import org.springframework.core.io.ClassPathResource;  
import org.springframework.core.io.Resource;  
public class Test {  
    public static void main(String[] args) {  
        Resource resource = new ClassPathResource("applicationContext.xml");
```

```

        BeanFactory factory = new XmlBeanFactory(resource);
        Student student = (Student)factory.getBean("studentbean");
        student.displayInfo();
    }
}

```

## 5. Create XML File.

**New Java Class**

Java Class

Type already exists.

Source folder: Spring\_Hello/src Browse...

Package: mypkg Browse...

☐ Enclosing type: Browse...

Name: Student

Modifiers: ☒ public ☐ package ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

? Finish Cancel

```

<?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:p="http://www.springframework.org/schema/p"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
</beans>

```



**Conclusion:**

## Practical No: 9

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

**Objective:** Develop a Java program to perform basic database operations, including inserting, updating, and deleting records from a specified table.

**Description:** This program focuses on interacting with a database table, implementing functionalities to insert new records, update existing records, and delete records as needed. It demonstrates the fundamental operations for managing data in a database table using Java.

**Program:**

### 1. Create a Table.

```
CREATE TABLE userid(  
id varchar2(30) NOT NULL PRIMARY KEY,  
pwd varchar2(30) NOT NULL, fullname varchar2(50),  
email varchar2(50)  
);
```

### 2. Connection to Database.

```
import java.sql.*;  
  
public class connect {  
    public static void main(String args[]) {  
  
        try {  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl",  
"login1", "pwd1");  
            if (con != null)  
                System.out.println("Connected");  
            else  
  
                System.out.println("Not  
Connected");  
            con.close();  
        } catch (Exception e) {  
  
            System.out.println(e);  
        }  
    }  
}
```

### 3. Insert Statement.

```
import java.sql.*;

public class insert1 {
    public static void main(String args[])
    {
        String id = "id1"; String pwd = "pwd1";
        String fullname = "geeks for geeks"; String email = "geeks@geeks.org";
        try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver"); Connection con = DriverManager.getConnection("
                jdbc:oracle:thin:@localhost:1521:orcl", "login1", "pwd1");
            Statement stmt = con.createStatement();

            // Inserting data in database
            String q1 = "insert into userid values('"+id+"', '"+pwd+"fullname+'", '"+email+"')";
            int x = stmt.executeUpdate(q1);
            if (x > 0)
                System.out.println("Successfully Inserted");
            else
                System.out.println("Insert Failed");
            con.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

Output: Successfully Registered.

### 4. Update Statement.

```
import java.sql.*;

public class update1
{
    public static void main(String args[])
    {
        String id = "id1"; String pwd = "pwd1";
        String newPwd = "newpwd"; try
        {
            Class.forName("oracle.jdbc.driver.OracleDriver"); Connection con = DriverManager.getConnection("
                jdbc:oracle:thin:@localhost:1521:orcl", "login1", "pwd1");
            Statement stmt = con.createStatement();
```

```

// Updating database
String q1 = "UPDATE userid set pwd = '" + newPwd +
'" WHERE id = '" + id + "' AND pwd = '" + pwd + "'";
int x = stmt.executeUpdate(q1);

if (x > 0)
    System.out.println("Password Successfully Updated");
else
    System.out.println("ERROR OCCURRED :(");
    con.close();
}
catch(Exception e)
{
    System.out.println(e);
}
}
}

```

Output: Password Successfully Updated

## 5. Delete Statement.

```

import java.sql.*;

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

        String id = "id2";
        String pwd = "pwd2";
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con = DriverManager.getConnection(" jdbc:oracle:thin:@localhost:1521:orcl",
"login1", "pwd1");
            Statement stmt = con.createStatement();

            // Deleting from database
            String q1 = "DELETE from userid WHERE id = '" + id + "' AND pwd = '" + pwd + "'";
            int x = stmt.executeUpdate(q1);
            if (x > 0)
                System.out.println("One User Successfully Deleted");

            else
                System.out.println("ERROR OCCURRED :(");

            con.close();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

```

```
}  
}
```

Output: One User Successfully Deleted

**Conclusion:**

## Practical No: 10

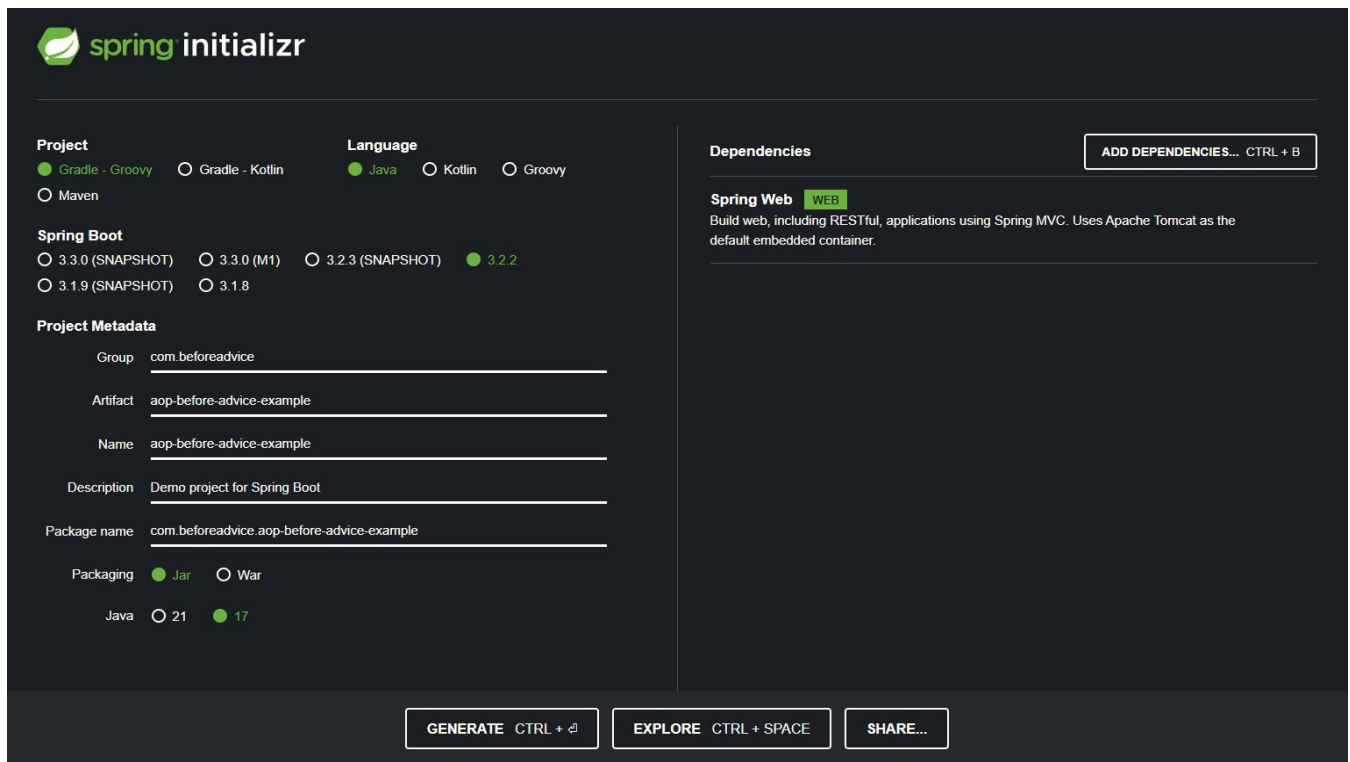
**Aim:** Write a program to demonstrate Spring AOP before advice.

**Objective:** Develop a Java program using the Spring framework to demonstrate the use of Aspect-Oriented Programming (AOP) with "before" advice.

**Description:** This program showcases the Spring AOP capability by implementing "before" advice, a type of cross-cutting concern that executes before the target method. It provides a clear example of how AOP can be used to modularize and manage cross-cutting concerns in an application.

### Program:

1. Open Spring Initializer <http://start.spring.io>
2. Provide the Group name: com.beforeadvice
3. Provide the Artifact Id: aop-before-advice-example
4. Add the Spring Web dependency.
5. Click on Generate button. Download and Extract it.



The screenshot shows the Spring Initializr web application interface. The header features the Spring logo and the text "spring initializr". The main content area is divided into several sections:

- Project:** Includes radio buttons for "Gradle - Groovy" (selected), "Gradle - Kotlin", and "Maven".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for "3.3.0 (SNAPSHOT)", "3.3.0 (M1)", "3.2.3 (SNAPSHOT)", "3.2.2" (selected), and "3.1.9 (SNAPSHOT)".
- Project Metadata:** Includes input fields for "Group" (com.beforeadvice), "Artifact" (aop-before-advice-example), "Name" (aop-before-advice-example), "Description" (Demo project for Spring Boot), and "Package name" (com.beforeadvice.aop-before-advice-example).
- Packaging:** Includes radio buttons for "Jar" (selected) and "War".
- Java:** Includes radio buttons for "21" and "17" (selected).
- Dependencies:** Includes a button "ADD DEPENDENCIES... CTRL + B" and a section for "Spring Web" with a "WEB" tag and a description: "Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container."

At the bottom, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...".



6. Import aop-before-advice-example Folder.
7. Add dependency for Spring AOP in 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.beforeadvice</groupId>
  <artifactId>aop-before-advice-example</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <packaging>jar</packaging>

  <name>aop-before-advice-example</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.2.RELEASE</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
    <!-- dependency for spring web -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <!-- added dependency for spring aop -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-aop</artifactId>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.springframework.boot</groupId>
```

```
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>

</project>
```

8. Create package com.beforeadvice.model and Add Student Model.

```
package com.beforeadvice.model;

public class Student {
    private String firstName;
    private String secondName;

    public Student() {}

    public String getFirstName() { return firstName; }public
    void setFirstName(String firstName)
    {
        this.firstName = firstName;
    }

    public String getSecondName() { return secondName; }

    public void setSecondName(String secondName)
    {
        this.secondName = secondName;
    }
}
```

9. Create package com.beforeadvice.service and Add StudentService class.

```
package com.beforeadvice.service;

import com.beforeadvice.model.Student;
import org.springframework.stereotype.Service;

@Service
public class StudentService {

    public Student addStudent(String fname, String sname)
    {
        System.out.println(
            "Add student service method called");Student
        stud = new Student(); stud.setFirstName(fname);
        stud.setSecondName(sname);
    }
}
```

```

        return stud;
    }
}

```

#### 10. Create package com.beforeadvice.controller and Add StudentController class

```

package com.beforeadvice.controller;

import com.beforeadvice.model.Student;
import com.beforeadvice.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired; import
org.springframework.web.bind.annotation.GetMapping; import
org.springframework.web.bind.annotation.RequestParam;import
org.springframework.web.bind.annotation.RestController;

@RestController
public class StudentController {

    @Autowired private StudentService studentService;

    @GetMapping(value = "/add")
    public Student addStudent(
        @RequestParam("firstName") String firstName,
        @RequestParam("secondName") String secondName)
    {
        return studentService.addStudent(firstName,
                                           secondName);
    }
}

```

#### 11. Create package com.beforeadvice.aspect and Add StudentServiceAspect class.

```

package com.beforeadvice.aspect;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class StudentServiceAspect {

    // the pointcut expression specifying execution of any
    // method in com.beforeadvice.service.StudentService
    // class of any return type with 0 or more number of
    // arguments
    @Pointcut("execution(* com.beforeadvice.service.StudentService.*(..)) ")
}

```

```
@Before("anyStudentService() && args(fname, sname)")
public void beforeAdvice(JoinPoint joinPoint,
                        String fname, String sname)
{
    System.out.println(
        "Before method:" + joinPoint.getSignature()+ "\n Adding Student
first name- "
        + fname + ", second name- " + sname);
}
```

```
{"firstName": "Harry", "secondName": "Potter"}
```

[illegible]

```

2022-02-20 13:55:56.279 INFO 8660 --- [main] c.b.AopBeforeAdviceExampleApplication : Starting AopBeforeAdviceExampleApplication on DESKTOP-QDGR1HJ with PID 8660 (C:\US
2022-02-20 13:55:56.281 INFO 8660 --- [main] c.b.AopBeforeAdviceExampleApplication : No active profile set, falling back to default profiles: default
2022-02-20 13:55:57.314 INFO 8660 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 9999 (http)
2022-02-20 13:55:57.324 INFO 8660 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-02-20 13:55:57.324 INFO 8660 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.29]
2022-02-20 13:55:57.437 INFO 8660 --- [main] o.s.c.r.c.c.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-02-20 13:55:57.437 INFO 8660 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1109 ms
2022-02-20 13:55:57.658 INFO 8660 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2022-02-20 13:55:57.879 INFO 8660 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 9999 (http) with context path ''
2022-02-20 13:55:57.884 INFO 8660 --- [main] c.b.AopBeforeAdviceExampleApplication : Started AopBeforeAdviceExampleApplication in 1.909 seconds (JVM running for 2.708)
2022-02-20 13:56:05.917 INFO 8660 --- [nio-9999-exec-1] o.a.c.c.c.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2022-02-20 13:56:05.917 INFO 8660 --- [nio-9999-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2022-02-20 13:56:05.923 INFO 8660 --- [nio-9999-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 6 ms
Before method:Student com.before_advice.service.StudentService.addStudent(String,String)
Adding Student with first name - Harry, second name - Potter
Add student service method called

```