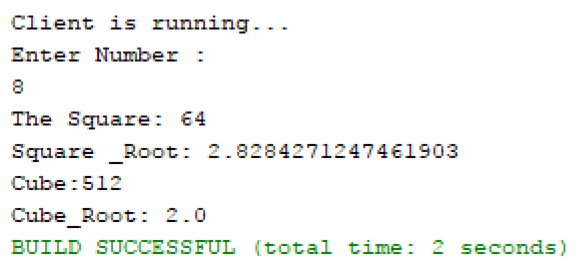
```
        catch (Exception e){ e.printStackTrace();
        }
    }
}
```

OUTPUT:



```
Client is running...
Enter Number :
8
The Square: 64
Square _Root: 2.8284271247461903
Cube:512
Cube_Root: 2.0
BUILD SUCCESSFUL (total time: 2 seconds)
```

Pratik Patil

Roll No: 58

KERALEEYA SAMAJAM(REGD.) DOMBIVLI’S
MODEL COLLEGE
EMPOWERED AUTONOMOUS

PRACTICAL 3

Date: 02-01-25

Aim: Write a program to show the object communication using RMI.

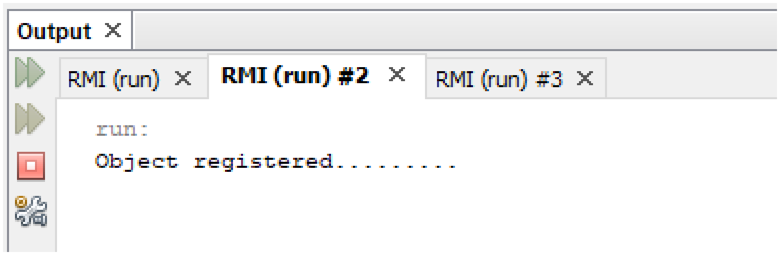
a) RMI-based application program to display current date and time.

```
import java.rmi.*;
public interface InterDate extends Remote {
public String display() throws Exception;
}
//Server code
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
public class ServerDate extends UnicastRemoteObject implements InterDate{ public
ServerDate() throws Exception
{
}
public String display() throws Exception
{
String str="";
Date d=new Date();
str=d.toString(); return
str;
}
public static void main(String args[]) throws Exception {
ServerDate s1=new ServerDate();
Naming.bind("DS", s1);
System.out.println("Object registered.............");
}
}
```

Client Code

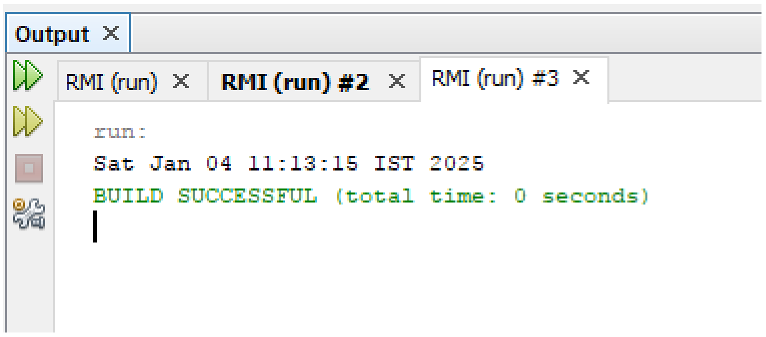
```
import java.rmi.*;
import java.io.*;
public class ClientDate {
public static void main(String args[]) throws Exception { String
s1;
InterDate h1=(InterDate)Naming.lookup("DS");
s1=h1.display();
System.out.println(s1);
}
}
```

OUTPUT:



Output ✕ RMI (run) ✕ RMI (run) #2 ✕ RMI (run) #3 ✕

```
run:
Object registered.........
```



Output ✕ RMI (run) ✕ RMI (run) #2 ✕ RMI (run) #3 ✕

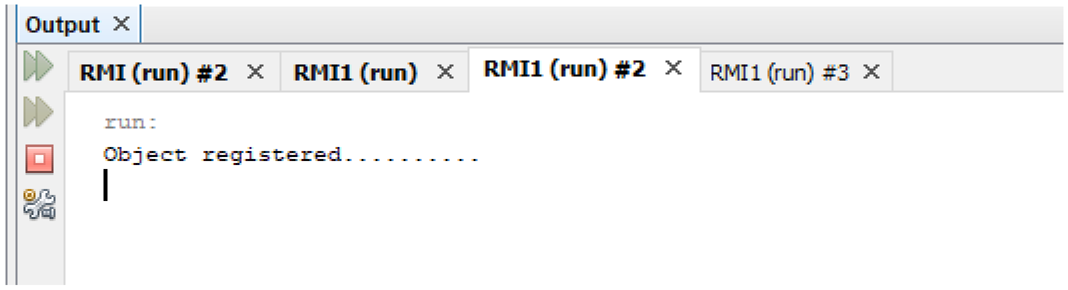
```
run:
Sat Jan 04 11:13:15 IST 2025
BUILD SUCCESSFUL (total time: 0 seconds)
```

b) RMI-based application program that converts digits to words. Eg 123 will be converted to ONE TWO THREE.

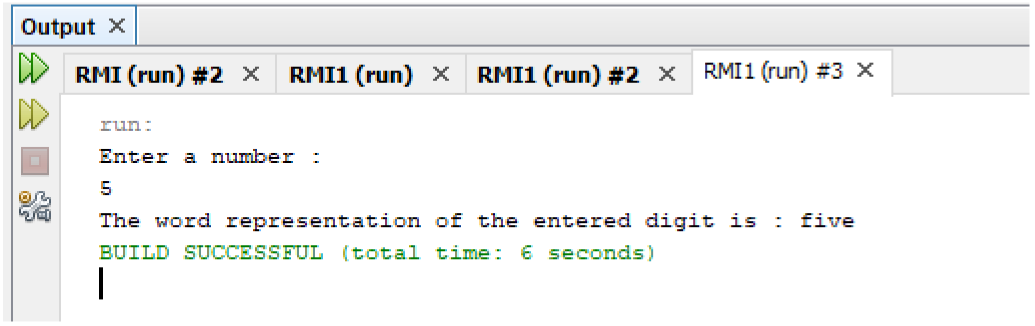
```
//interface.java
import java.rmi.*;
public interface InterConvert extends Remote{
public String convertDigit(String no) throws Exception;
}
//server code
import java.rmi.*;
import java.rmi.server.*;
public class ServerConvert extends UnicastRemoteObject implements InterConvert{ public
ServerConvert() throws Exception
{
}
public String convertDigit(String no) throws Exception
{
String str="";
for(int i=0;i<no.length();i++)
{
int p=no.charAt(i);
if(p==48)
{
str+="zero";
}
if(p==49)
{
str+="one";
}
if(p==50)
{
str+="two";
}
if(p==51)
{
str+="three";
}
if(p==52)
{
str+="four";
}
if(p==53)
{
str+="five";
}
if(p==54)
{
str+="six";
}
if(p==55)
{
str+="seven";
}
if(p==56)
{
str+="eight";
}
if(p==57)
{
str+="nine";
}
}
return str;
}
public static void main(String args[]) throws Exception {
```

```
ServerConvert s1=new ServerConvert(); Naming.bind("Wrd",
s1);
System.out.println("Object registered ............... ");
}
}
Client code import
java.rmi.*; import
java.io.*;
public class ClientConvert {
public static void main(String args[]) throws Exception
{
InterConvert h1=(InterConvert)Naming.lookup("Wrd");
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter a number : \t");
String no=br.readLine();
String ans=h1.convertDigit(no);
System.out.println("The word representation of the entered digit is : "+ans);
}
}
```

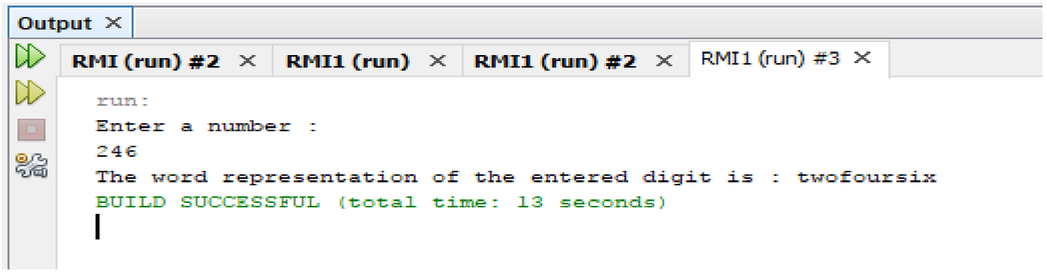
<u>OUTPUT:</u>



```
run:
Object registered..........
```



```
run:
Enter a number :
5
The word representation of the entered digit is : five
BUILD SUCCESSFUL (total time: 6 seconds)
```



```
run:
Enter a number :
246
The word representation of the entered digit is : twofoursix
BUILD SUCCESSFUL (total time: 13 seconds)
```

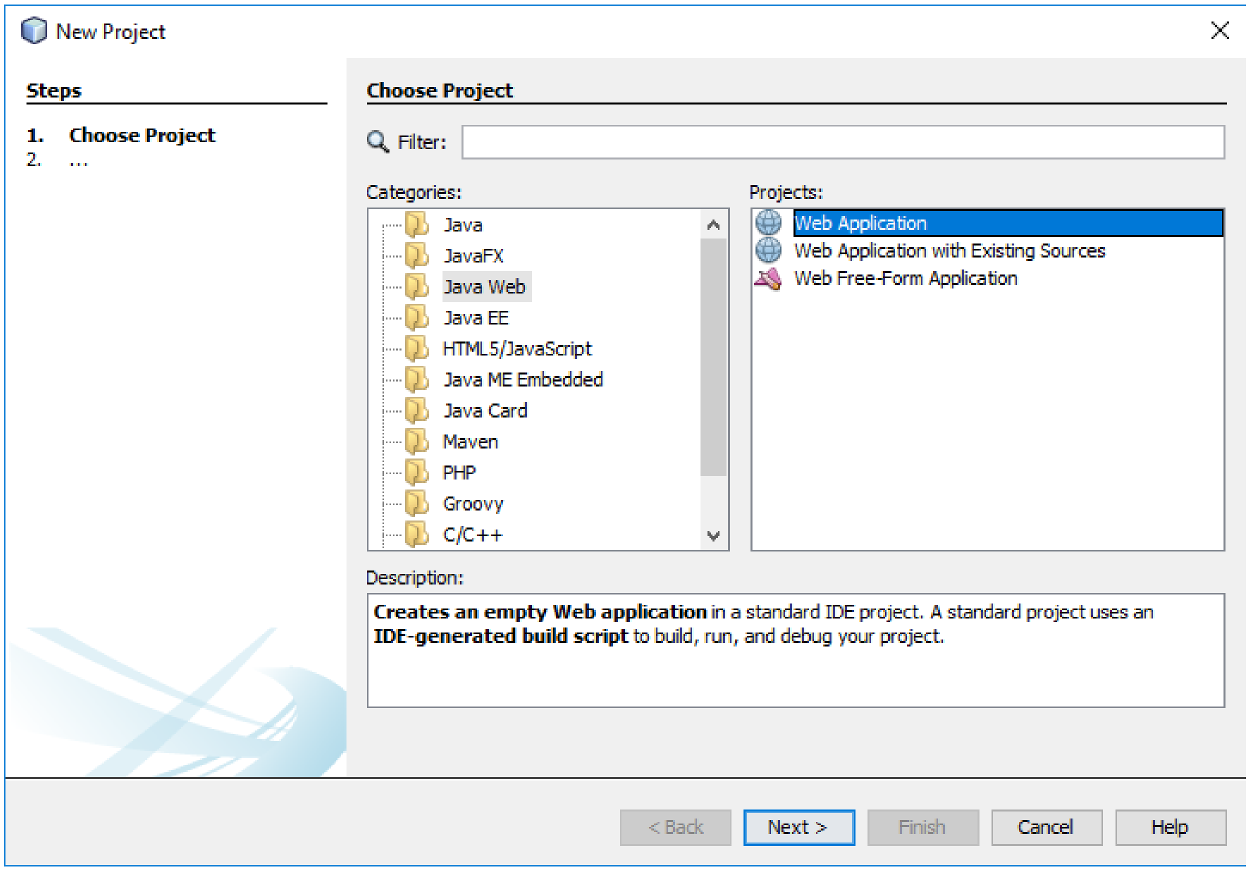
PRACTICAL 4

Date: 16-01-25

Aim: Show the implementation of webservices.

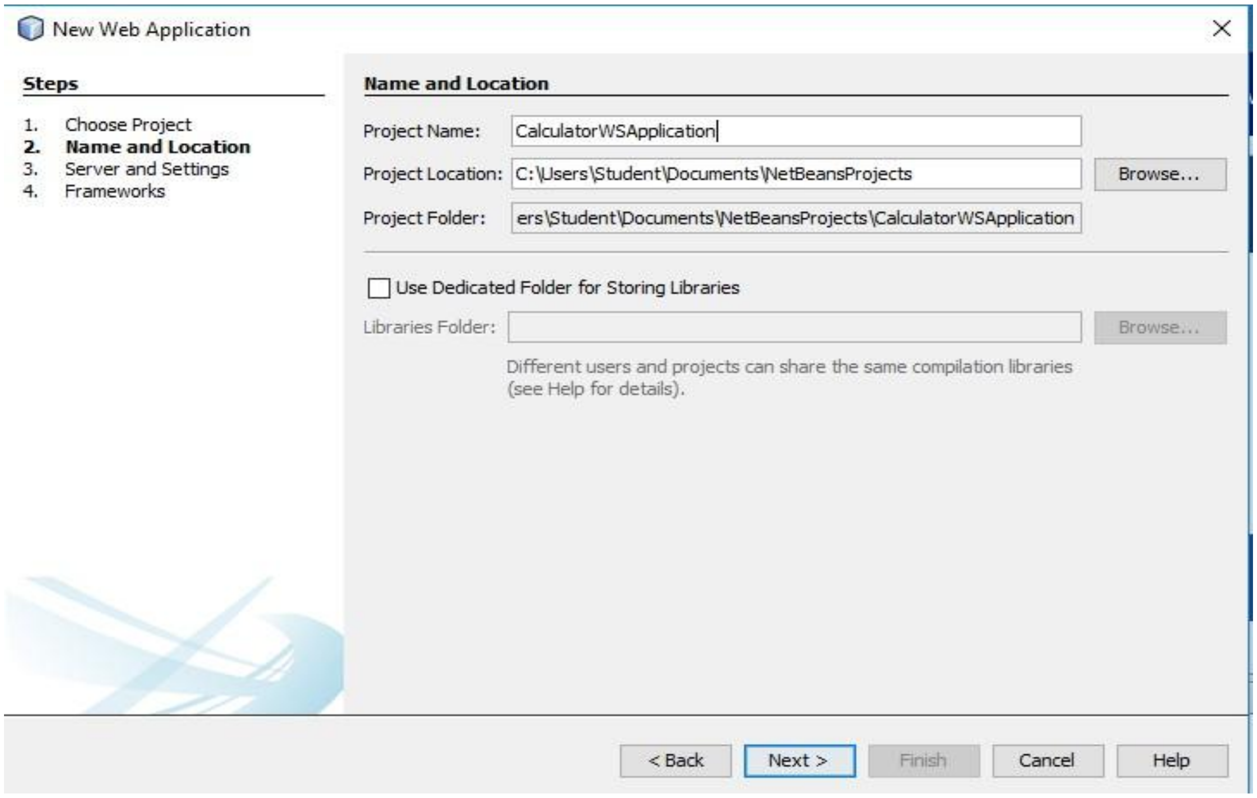
a) Implement Big web services.

Step 1: Select File → New Project → Java Web → Web Application.



Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

Step 2: Name the project as “CalculatorWSApplication”.Click on Next.



New Web Application

Steps

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

Name and Location

Project Name: CalculatorWSApplication

Project Location: C:\Users\Student\Documents\NetBeansProjects

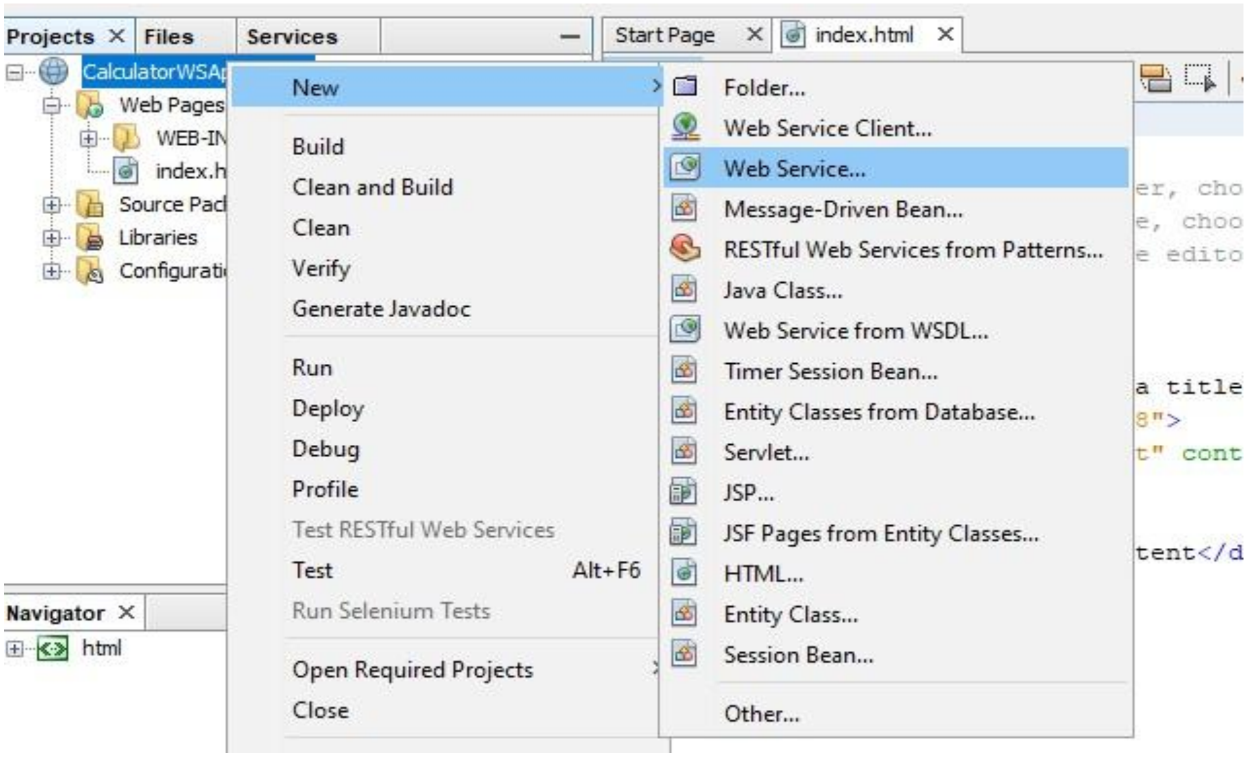
Project Folder: ers\Student\Documents\NetBeansProjects\CalculatorWSApplication

Use Dedicated Folder for Storing Libraries

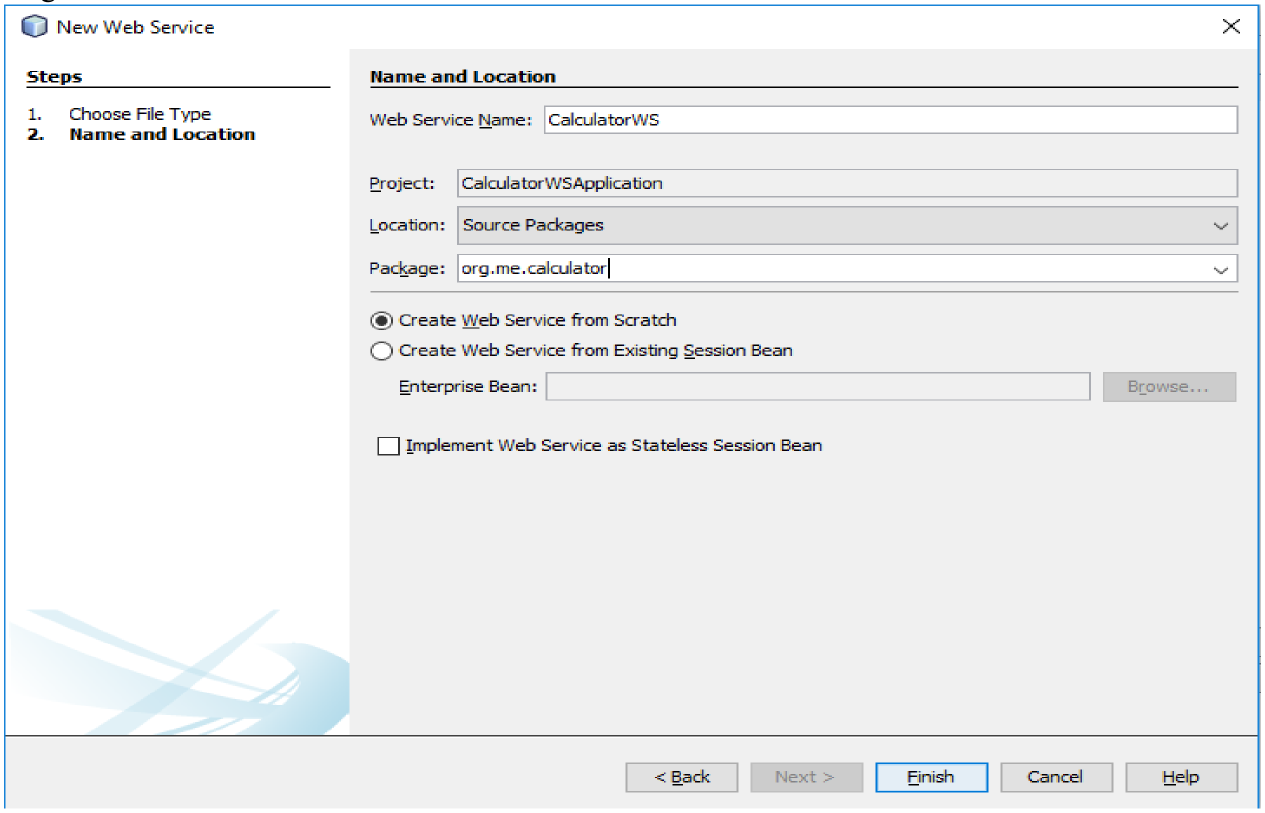
Libraries Folder:

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

Step 3: A window like this will appear right click on the project that we created New → Web Service.



Step 4: Give the name as "CalculatorWS" and also provide a package name as "org.me.calculator".Click on Finish.



Web Service Name: CalculatorWS

Project: CalculatorWSApplication

Location: Source Packages

Package: org.me.calculator

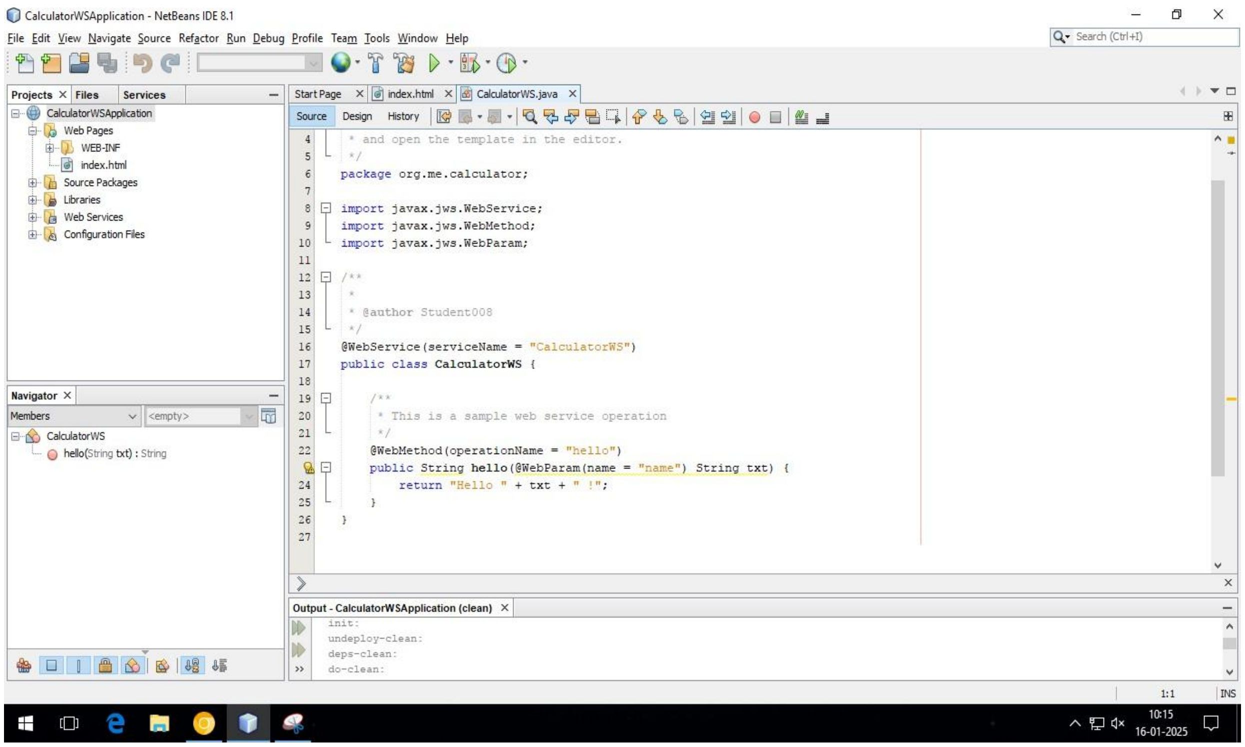
(●) Create Web Service from Scratch

( ) Create Web Service from Existing Session Bean

Enterprise Bean:

[ ] Implement Web Service as Stateless Session Bean

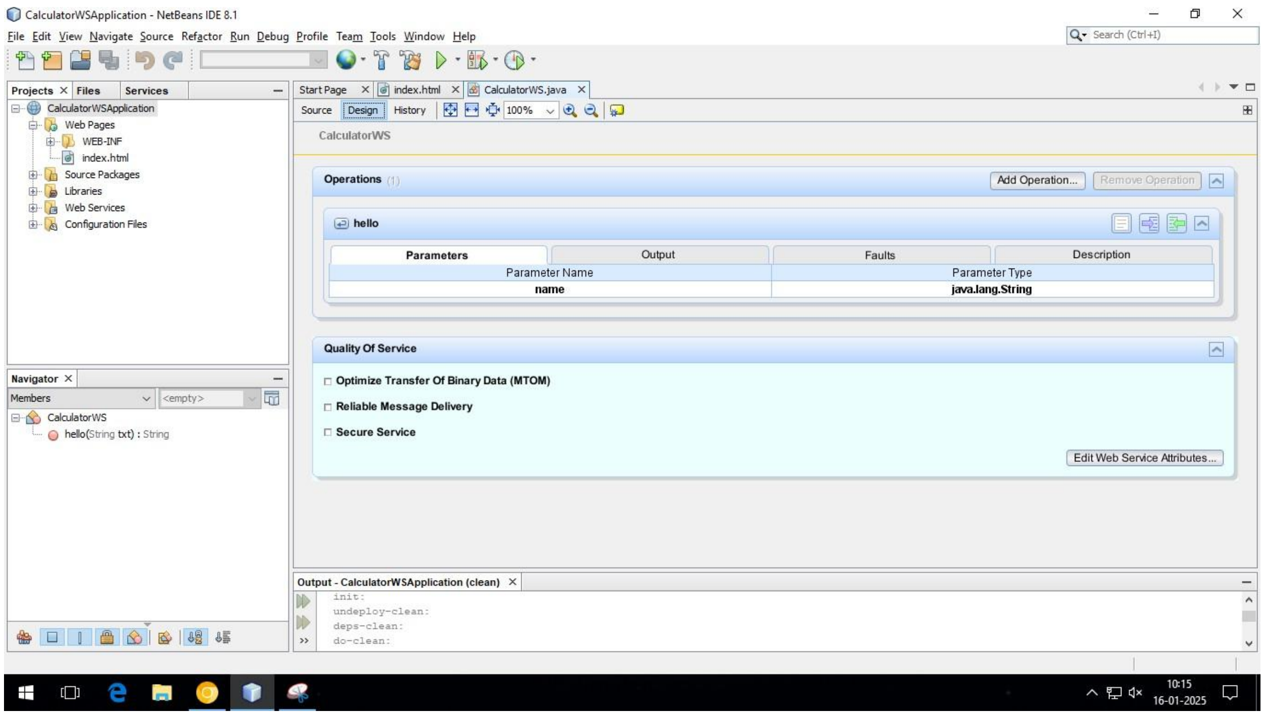
Step 5: Now a window like this will appear along with the source code as below.



```
 * and open the template in the editor.
 */
package org.me.calculator;

import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author Student008
 */
@WebService(serviceName = "CalculatorWS")
public class CalculatorWS {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}
```

```
init:
undeploy-clean:
deps-clean:
do-clean:
```

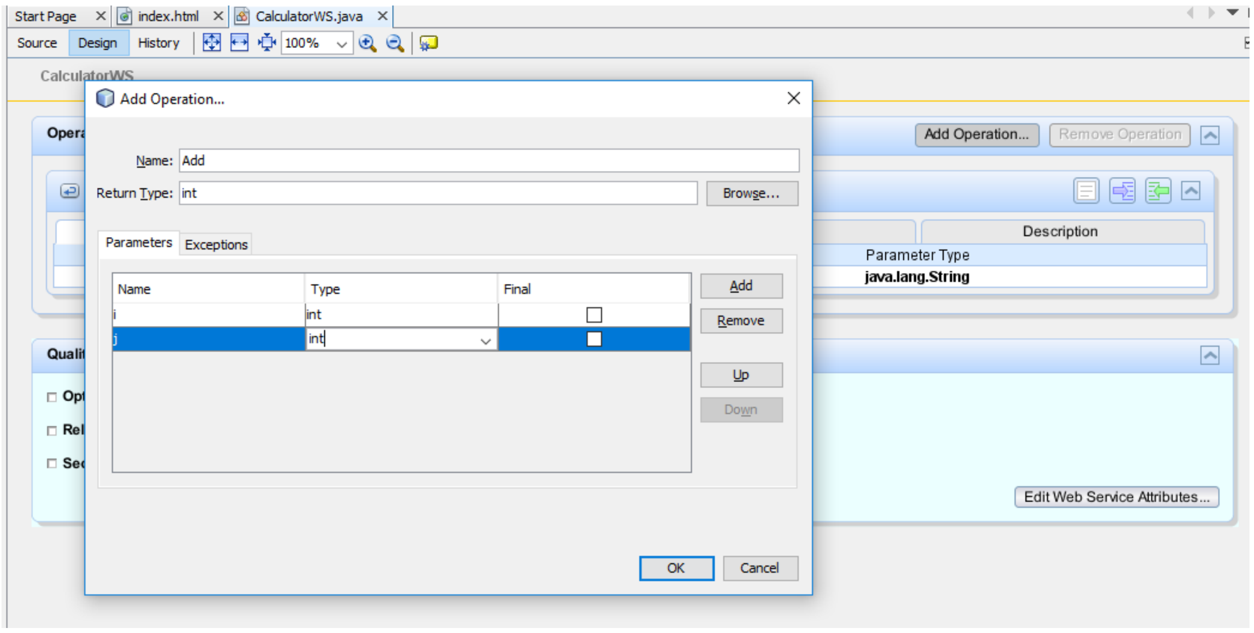
Step 6: Go to the Design tab as shown below.


CalculatorWSApplication - NetBeans IDE 8.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Files Services
CalculatorWSApplication
Web Pages
WEB-INF
index.html
Source Packages
Libraries
Web Services
Configuration Files
Start Page index.html CalculatorWS.java
Source Design History 100%
CalculatorWS
Operations (1) Add Operation... Remove Operation
hello
Parameters Output Faults Description
Quality Of Service
Optimize Transfer Of Binary Data (MTOM)
Reliable Message Delivery
Secure Service
Edit Web Service Attributes...
Navigator Members <empty>
CalculatorWS
hello(String txt) : String
Output - CalculatorWSApplication (clean)
10:15
16-01-2025

Parameter Name	Parameter Type
name	java.lang.String

```
init:
undeploy-clean:
deps-clean:
do-clean:
```

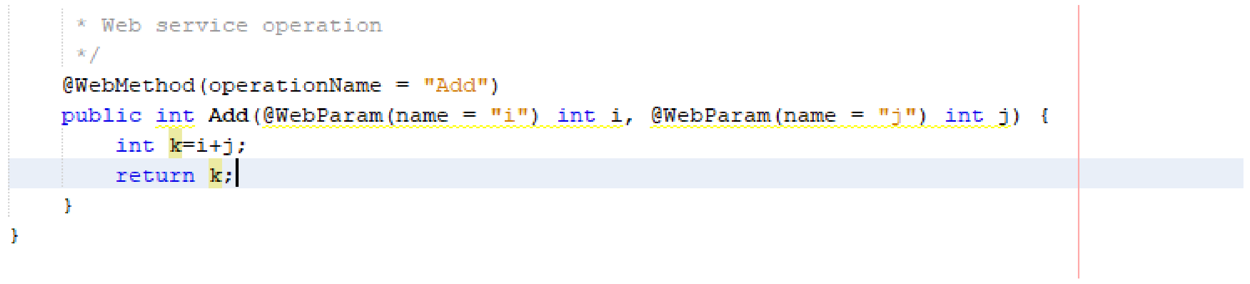
Step 7: In the first dialog box give the Name as “Add” and Return Type as “int”. Then click on the add button to add “i” and “j” parameters and provide the type as “int”. Click on Ok.


Start Page index.html CalculatorWS.java
Source Design History 100%
CalculatorWS
Add Operation...
Parameters Exceptions
Browse...
Add
Remove
Up
Down
OK
Cancel
Add Operation... Remove Operation
Description
Parameter Type
java.lang.String
Edit Web Service Attributes...

Name: Add
Return Type: int

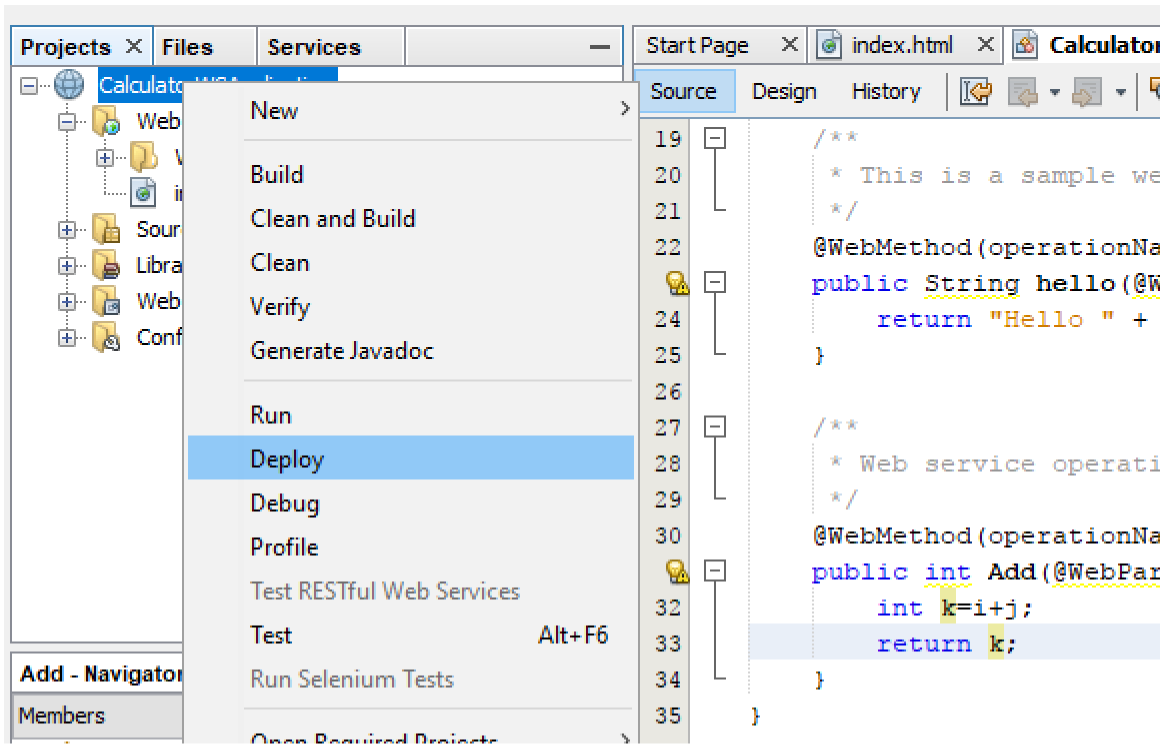
Name	Type	Final
i	int	☐
j	int	☐

Step 8: The following code will be visible in the source code .



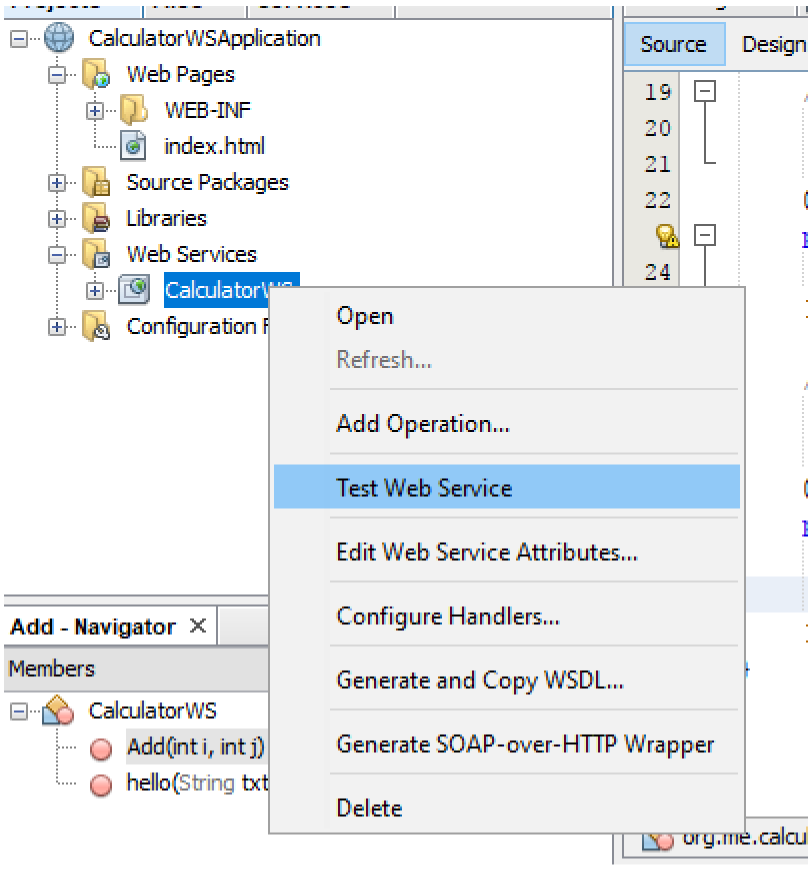
```
     * Web service operation
     */
    @WebMethod(operationName = "Add")
    public int Add(@WebParam(name = "i") int i, @WebParam(name = "j") int j) {
        int k=i+j;
        return k;
    }
}
```

Step 9: Now Right Click on the project and select "Deploy".

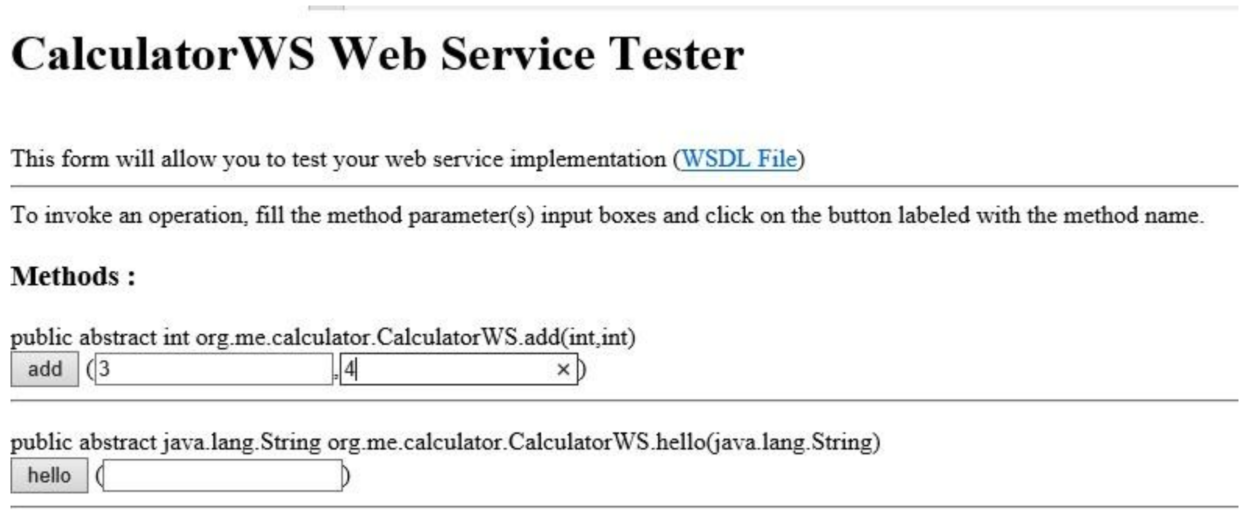


```
    /**
     * This is a sample we
     */
    @WebMethodoperationNa
    public String hello(@W
        return "Hello " +
    }

    /**
     * Web service operati
     */
    @WebMethod(operationNa
    public int Add(@WebPar
        int k=i+j;
        return k;
    }
}
```

Step 10: Now go to the project tab→web services→CalculatorWS. Right click on it and select "Test Web Services".



Step 11: A window like this will appear provide the numbers as "3" and "4" click on the add button.



CalculatorWS Web Service Tester

This form will allow you to test your web service implementation (WSDL File)

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

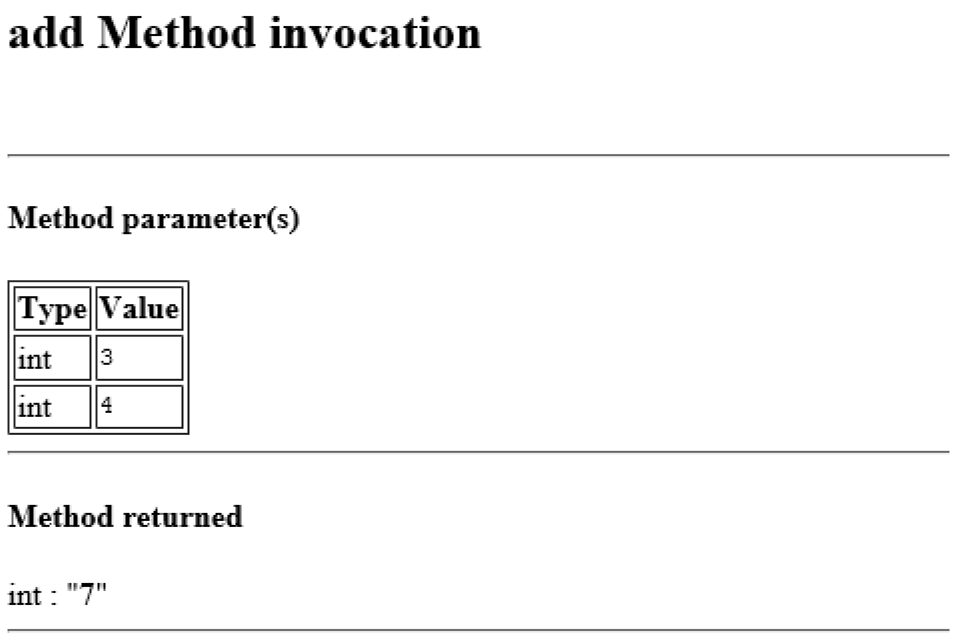
public abstract int org.me.calculator.CalculatorWS.add(int,int)

add (3 ,4)

public abstract java.lang.String org.me.calculator.CalculatorWS.hello(java.lang.String)

hello ( )

Step 12: Here we can see the method returned as "int 7".



add Method invocation

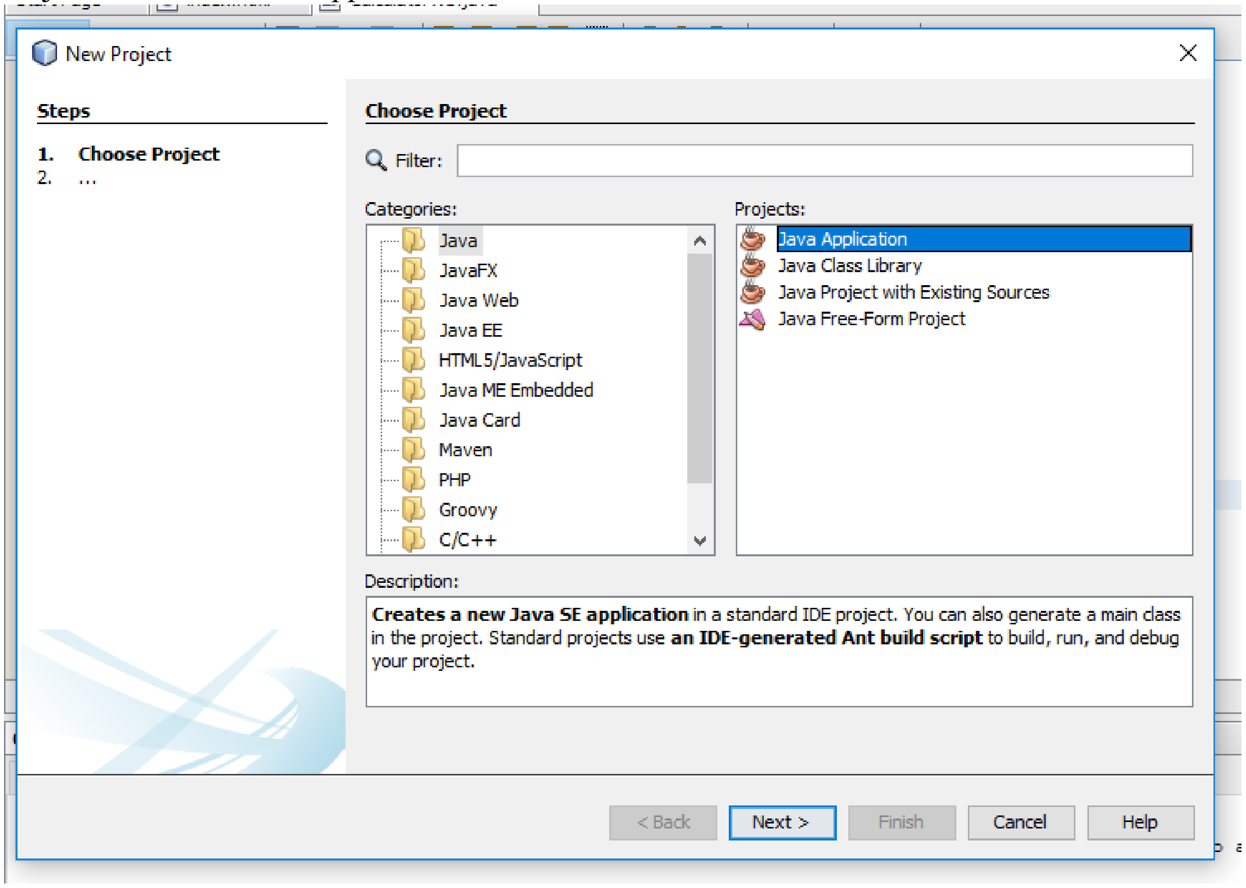
Method parameter(s)

Type	Value
int	3
int	4

Method returned

int : "7"

Step 13: Now we have to create a client for consuming the web services . Click on File → New Project → Java → Java Application.



New Project

Steps

1. **Choose Project**
2. ...

Choose Project

Filter:

Categories:

- Java
- JavaFX
- Java Web
- Java EE
- HTML5/JavaScript
- Java ME Embedded
- Java Card
- Maven
- PHP
- Groovy
- C/C++

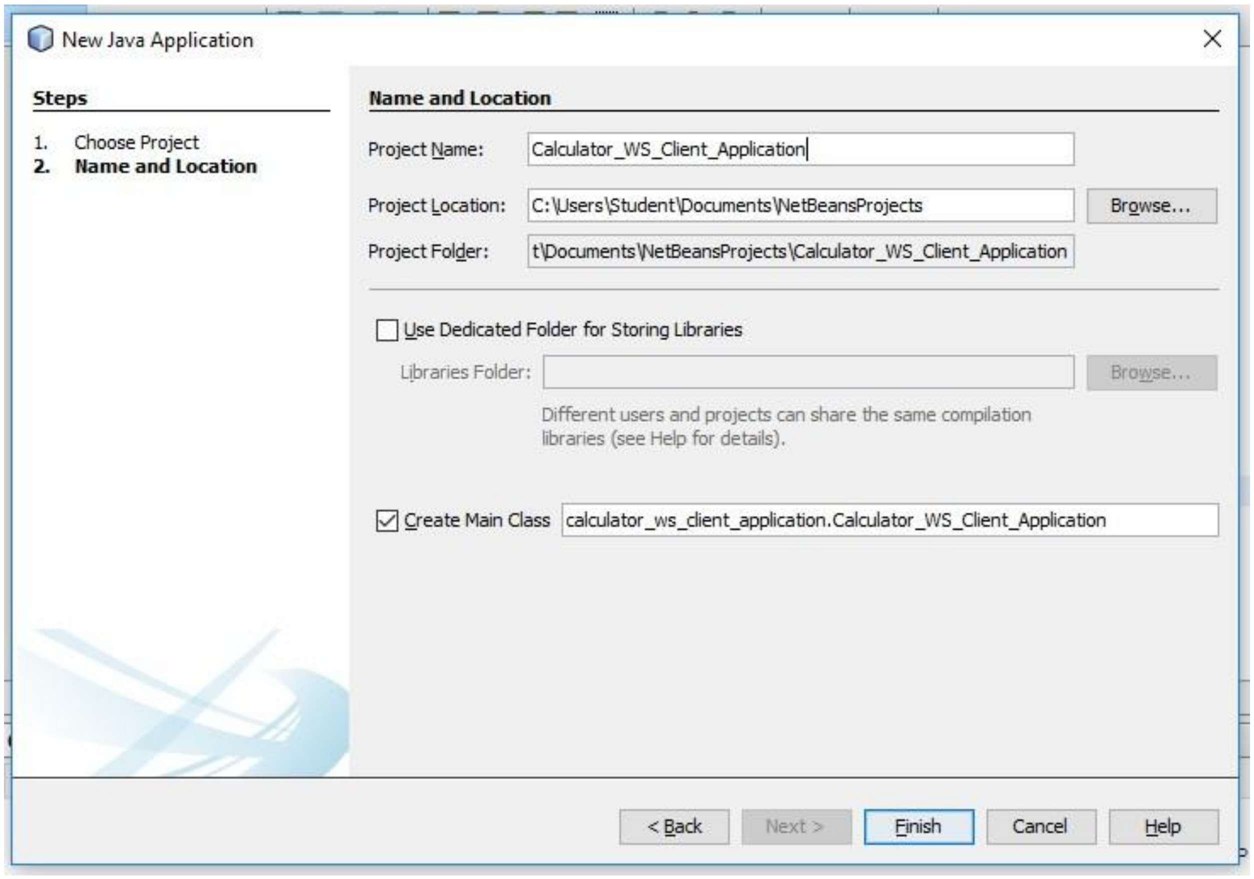
Projects:

- Java Application
- Java Class Library
- Java Project with Existing Sources
- Java Free-Form Project

Description:

Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

Step 14: Give the Name as "Calculator_WS_Client_Application. Click on Finish.



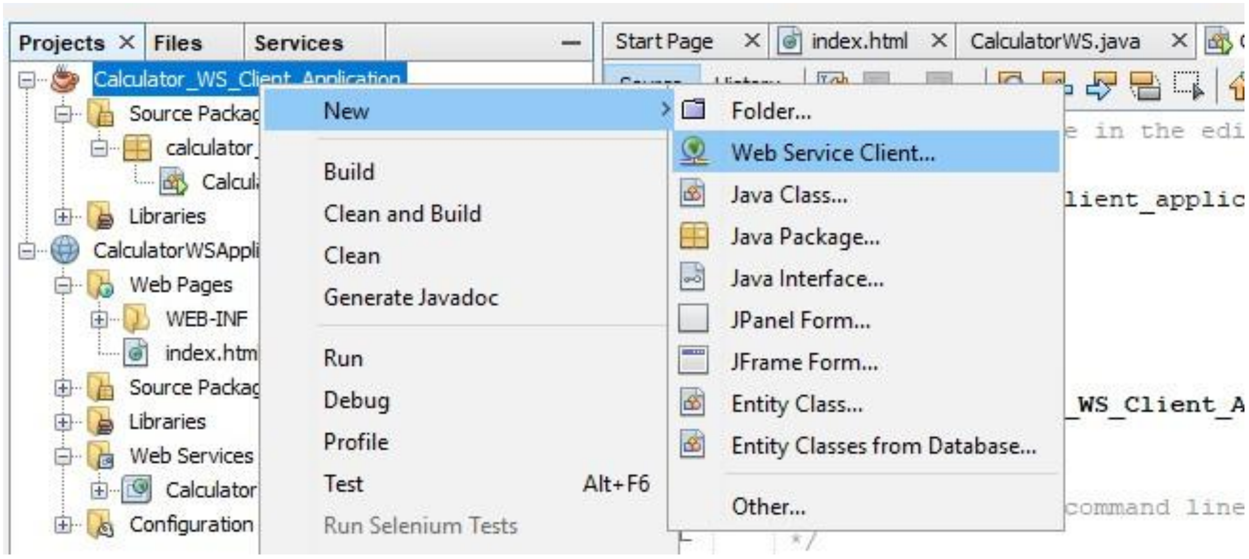
Project Name: Calculator_WS_Client_Application

Project Location: C:\Users\Student\Documents\NetBeansProjects

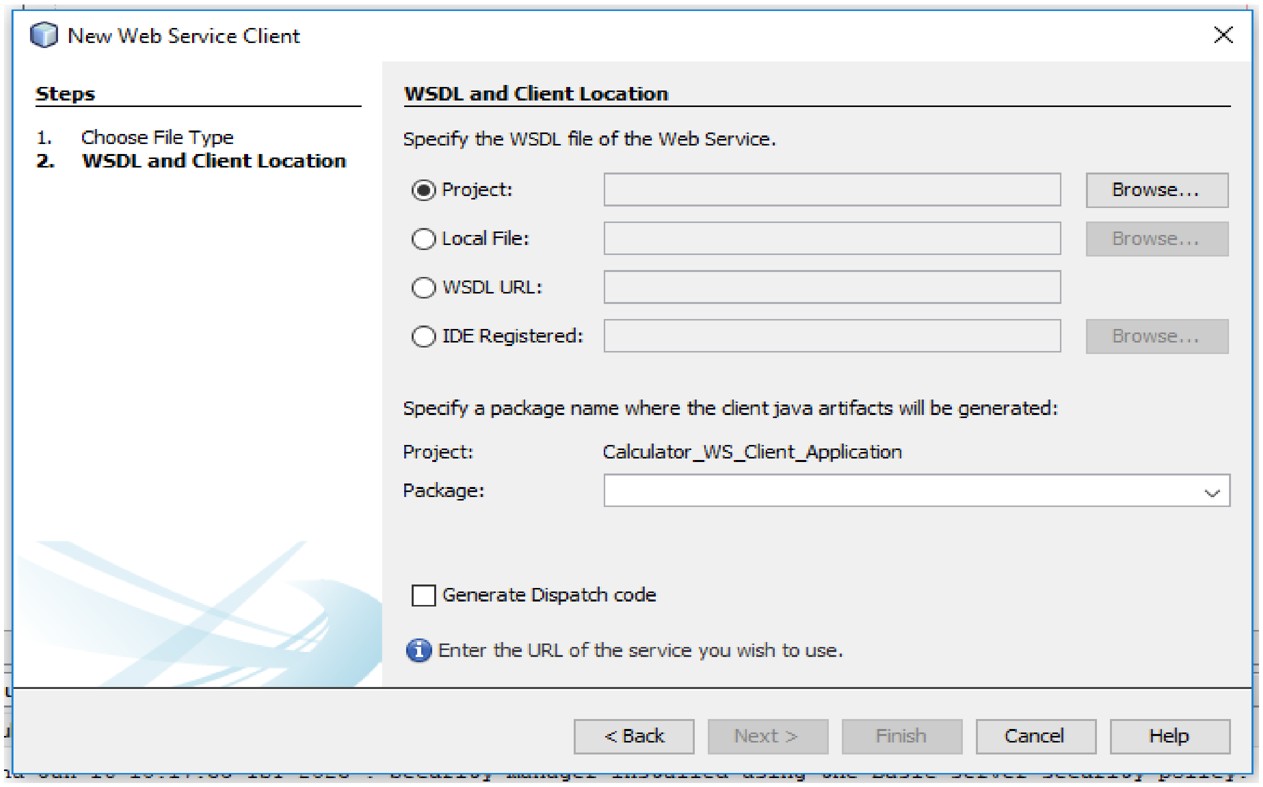
Project Folder: t\Documents\NetBeansProjects\Calculator_WS_Client_Application

Create Main Class: calculator_ws_client_application.Calculator_WS_Client_Application

Step 15: Right click on the project New → Web Service → Client.



Step 16: Select project as the WSDL File of the Web Service and click on Browse.



New Web Service Client

Steps

1. Choose File Type
2. **WSDL and Client Location**

WSDL and Client Location

Specify the WSDL file of the Web Service.

(●) Project: [ ] Browse...
( ) Local File: [ ] Browse...
( ) WSDL URL: [ ]
( ) IDE Registered: [ ] Browse...

Specify a package name where the client java artifacts will be generated:

Project: Calculator_WS_Client_Application

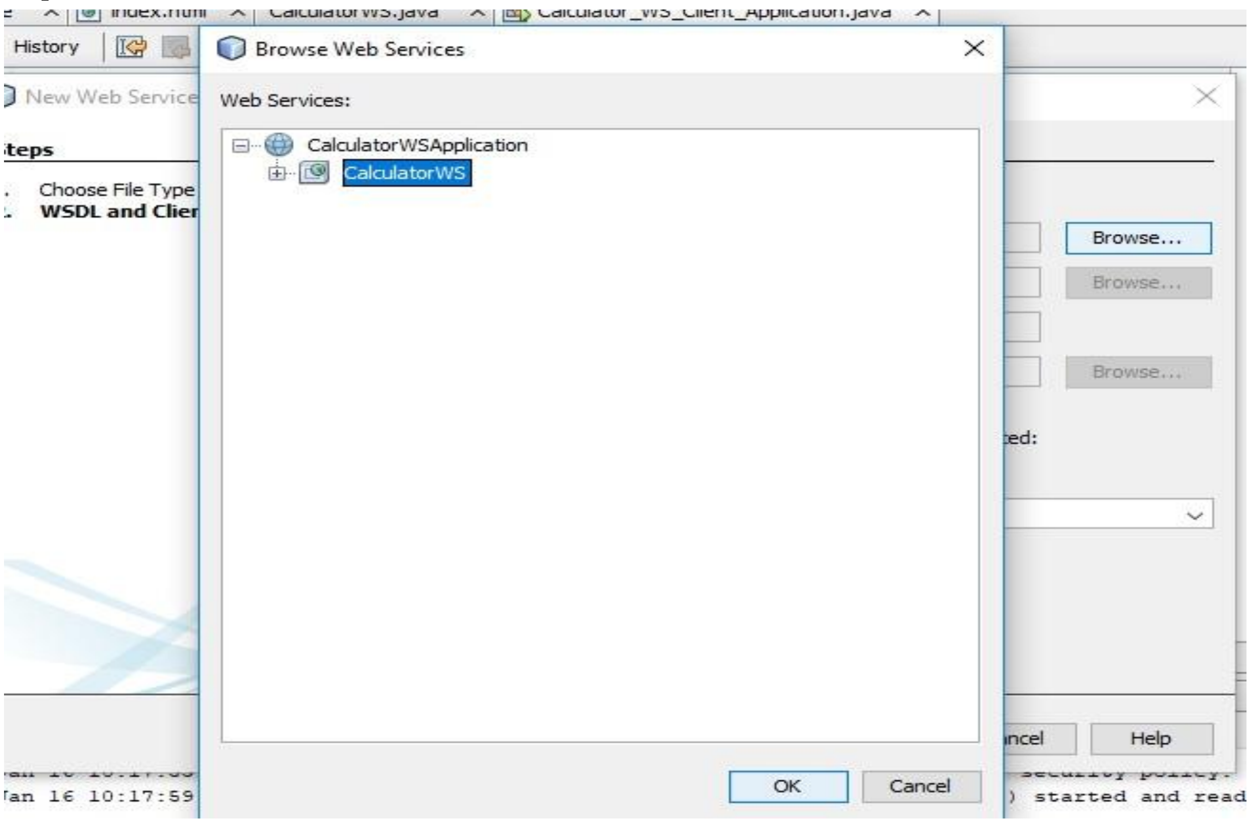
Package: [ ]

[ ] Generate Dispatch code

Enter the URL of the service you wish to use.

< Back | Next > | Finish | Cancel | Help

Step 17: Select CalculatorWS. Click on Ok.



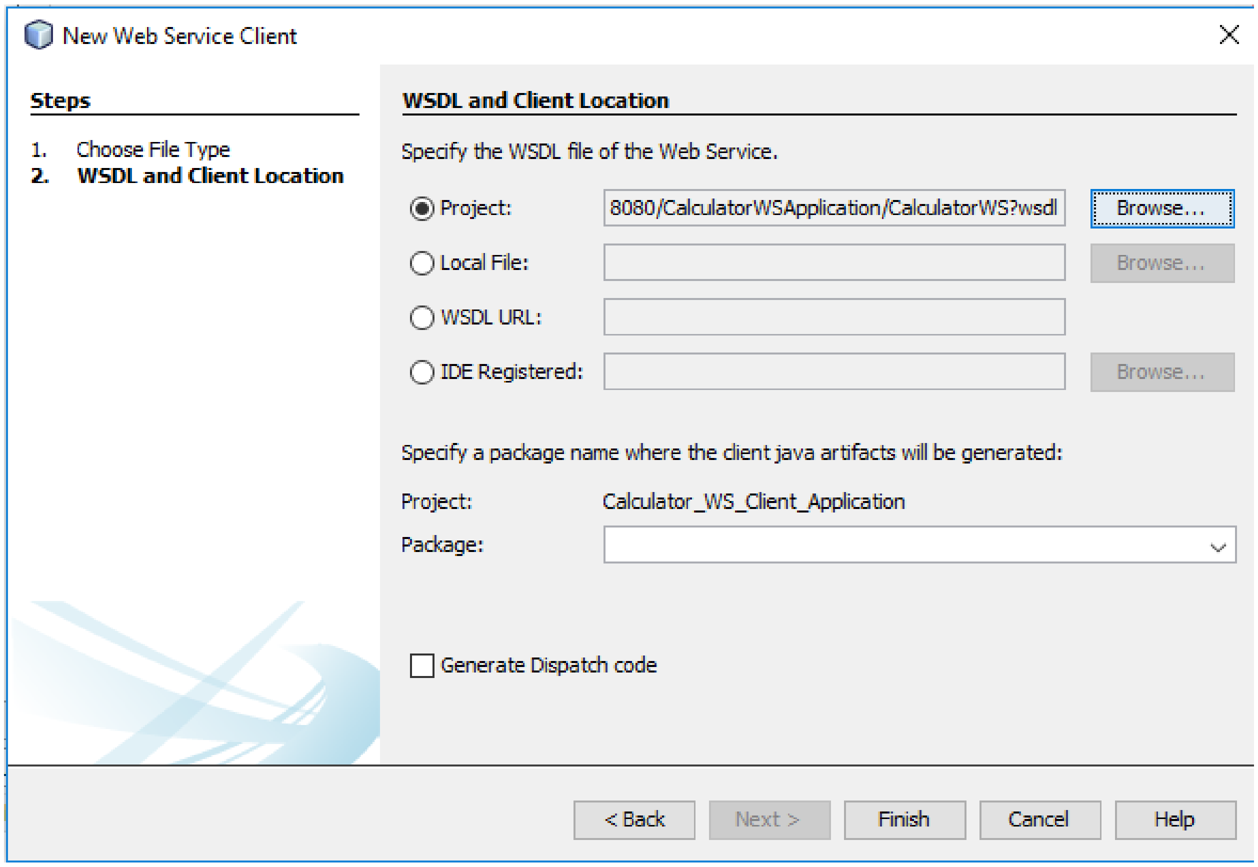
Browse Web Services

Web Services:

- CalculatorWSApplication
  - CalculatorWS

OK | Cancel

Step 18: A window like this will appear .



Specify the WSDL file of the Web Service.

(●) Project: 8080/CalculatorWSApplication/CalculatorWS?wsdl

( ) Local File:

( ) WSDL URL:

( ) IDE Registered:

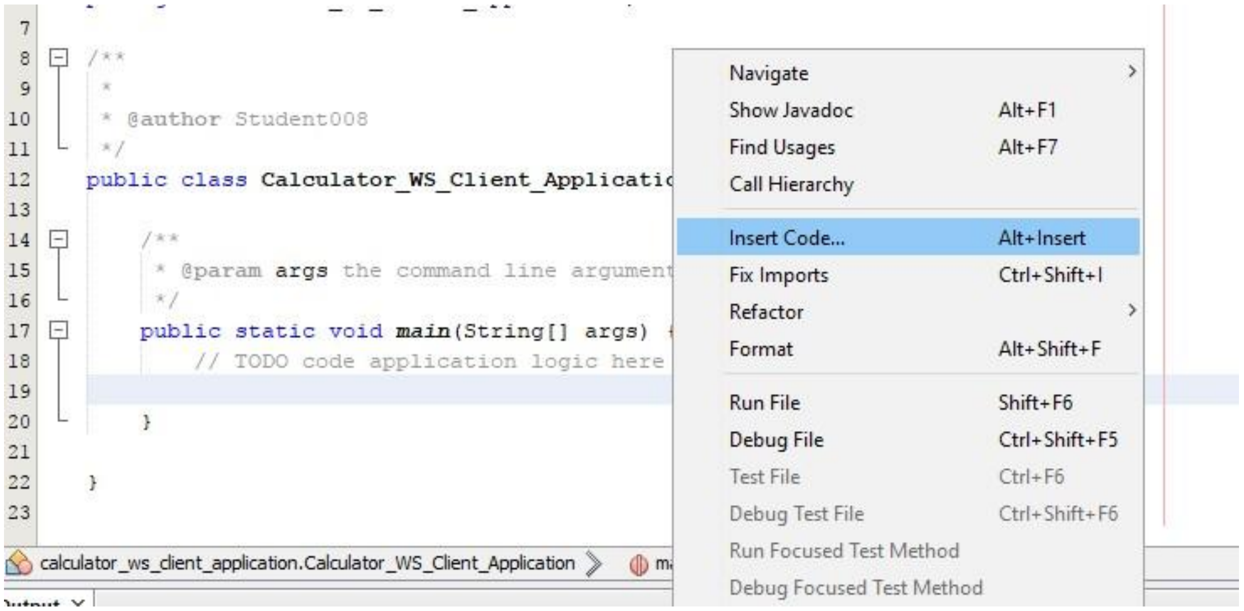
Specify a package name where the client java artifacts will be generated:

Project: Calculator_WS_Client_Application

Package:

☐ Generate Dispatch code

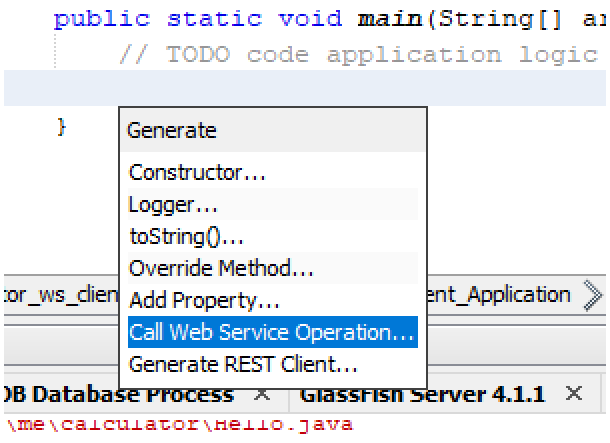
Step 17: Right click on the main method area and select “Insert Code”.


```
/**
 *
 * @author Student008
 */
public class Calculator_WS_Client_Applicati

    /**
     * @param args the command line argument
     */
    public static void main(String[] args) {
        // TODO code application logic here

    }

}
```

Pratik Patil

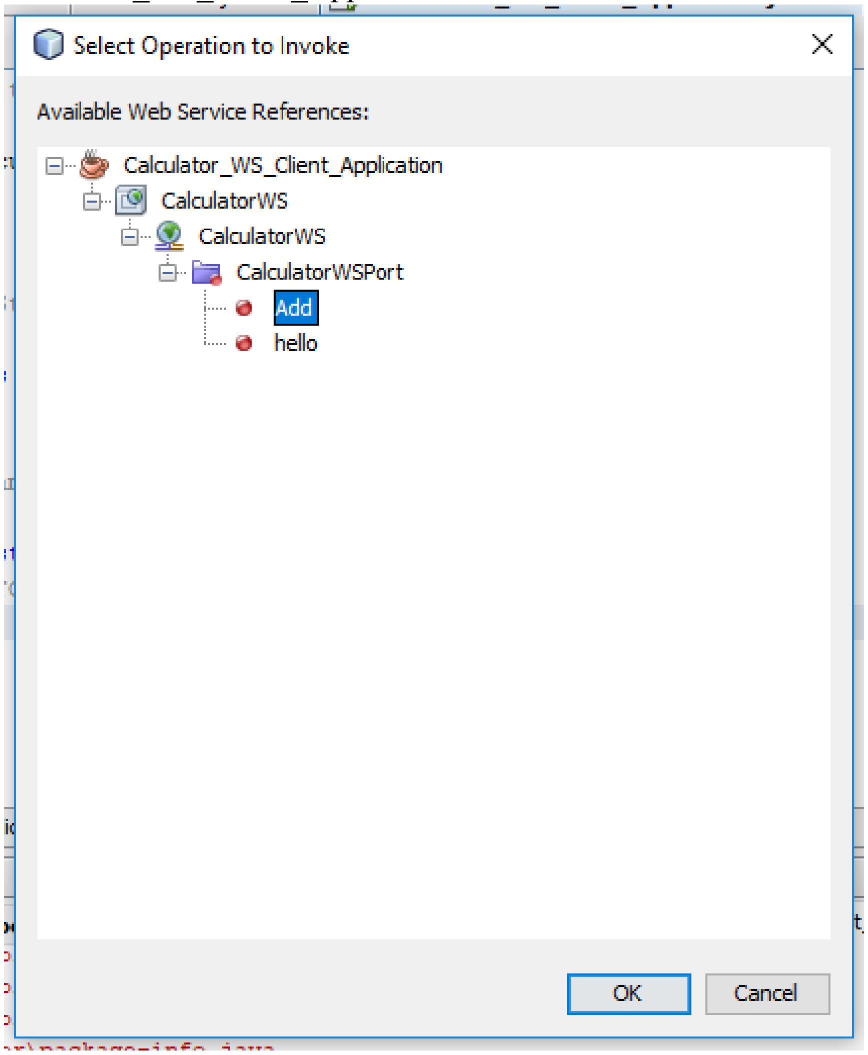
Roll No: 58

Step 18: Select on the “Call Web Service Operation”.



```
public static void main(String[] a
    // TODO code application logic

}
```

Step 19: Select Calculator_WS_Client_Application→CalculatorWS→CalculatorWSPort→Add.Click on Ok.

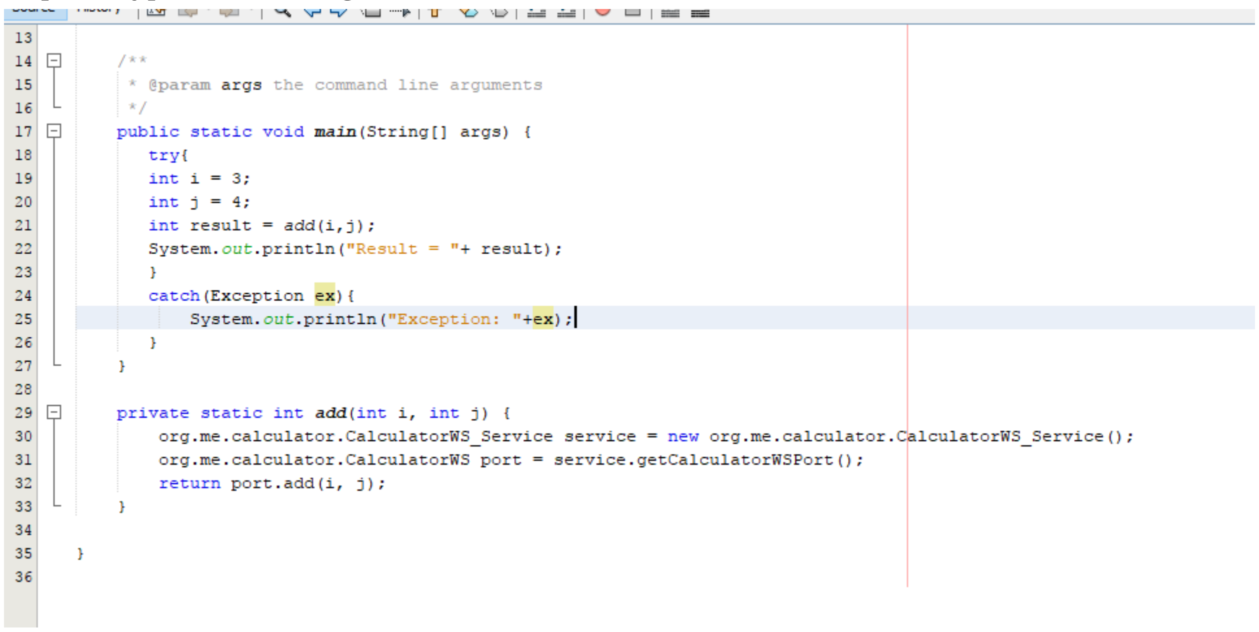


Select Operation to Invoke
Available Web Service References:
Calculator_WS_Client_Application
CalculatorWS
CalculatorWS
CalculatorWSPort
Add
hello
OK
Cancel

Pratik Patil

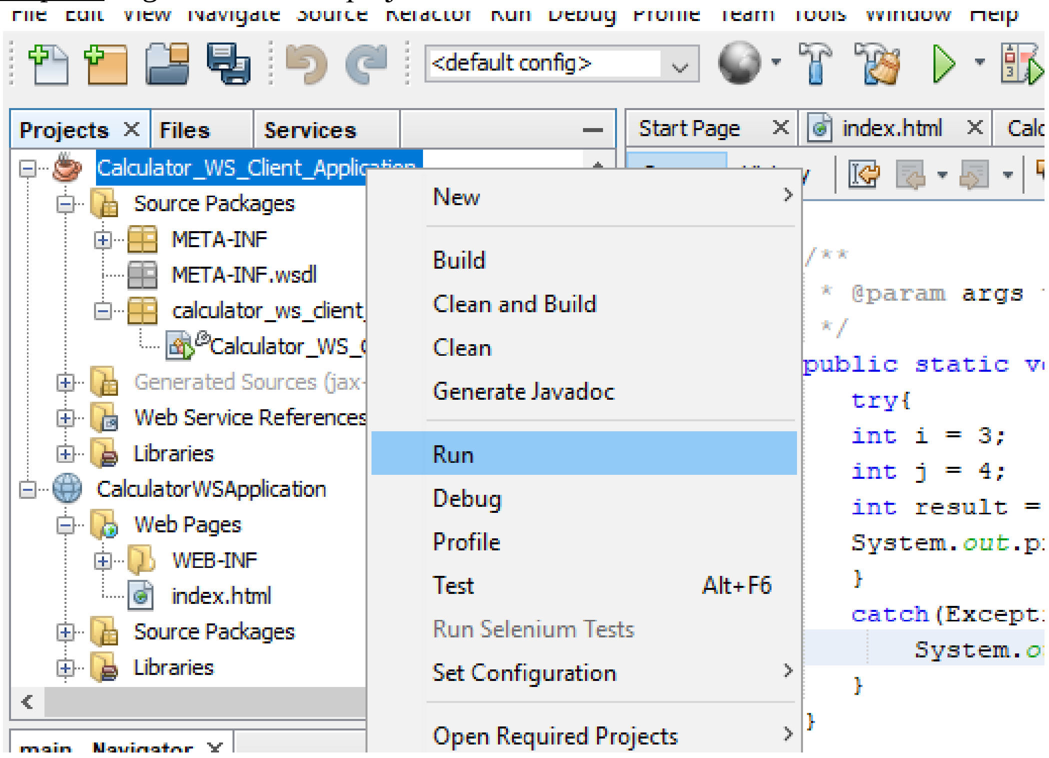
Roll No: 58

Step 20: Type the following code as shown below.



```
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        try{
        int i = 3;
        int j = 4;
        int result = add(i,j);
        System.out.println("Result = "+ result);
        }
        catch(Exception ex){
            System.out.println("Exception: "+ex);
        }
    }

    private static int add(int i, int j) {
        org.me.calculator.CalculatorWS_Service service = new org.me.calculator.CalculatorWS_Service();
        org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();
        return port.add(i, j);
    }

}
```

Step 21: Right click on the project→Run.


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects
Files
Services
Start Page
index.html
Calculator_WS_Client_Application
Source Packages
META-INF
META-INF.wsdl
calculator_ws_client
Calculator_WS_
Generated Sources (jax-
Web Service References
Libraries
CalculatorWSApplication
Web Pages
WEB-INF
index.html
Source Packages
Libraries
New
Build
Clean and Build
Clean
Generate Javadoc
Run
Debug
Profile
Test Alt+F6
Run Selenium Tests
Set Configuration
Open Required Projects
main Navigator