EXPERIMENT NO. 5

Title:
To create a simple web service and write any distributed application to consume the web service.

🎯 Objective:
By the end of this assignment, the student will be able to:
- Create, deploy, and test a Web Service application using Java (JAX-WS).
- Consume the deployed Web Service using a client application.

🧰 Tools Required:
- **NetBeans IDE** (preferably 12+)
- **GlassFish Server** (4.1 or later)
- **Java SE 8 / Java EE 7**

⚙️ Web Service Implementation (Server Side)
✅ Step 1: Create Web Application
- Open **NetBeans → File → New Project → Java Web → Web Application**
- Project Name: CalculatorWSApplication
- Server: **GlassFish Server**
- Finish.

✅ Step 2: Create the Web Service
Right-click the Project → New → Web Service
- Name: CalculatorWS
- Package: org.me.calculator
- Click *Finish.*

Replace the default hello() method with the following:

📄 Source Code — CalculatorWS.java
```
package org.me.calculator;

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

@WebService(serviceName = "CalculatorWS")
public class CalculatorWS {

    @WebMethod(operationName = "add")
    public int add(@WebParam(name = "num1") int num1, @WebParam(name = "num2") int
num2) {
        return num1 + num2;
    }

    @WebMethod(operationName = "subtract")
    public int subtract(@WebParam(name = "num1") int num1, @WebParam(name = "num2")
int num2) {
        return num1 - num2;
```

```
    }

    @WebMethod(operationName = "multiply")
    public int multiply(@WebParam(name = "num1") int num1, @WebParam(name =
"num2") int num2) {
        return num1 * num2;
    }

    @WebMethod(operationName = "divide")
    public double divide(@WebParam(name = "num1") double num1, @WebParam(name =
"num2") double num2) {
        if (num2 == 0)
            throw new IllegalArgumentException("Division by zero is not allowed.");
        return num1 / num2;
    }
}
```

✅ **Step 3: Deploy & Test**

- Right-click project → **Deploy**
  (GlassFish server will start and deploy the service)
- In **Projects** view → expand Web Services → right-click CalculatorWS → choose **Test Web Service**.

This opens a browser window with a **test client GUI**.
Try entering numbers for each operation and check the response.

🌐 **WSDL (Auto Generated)**

Once deployed, your WSDL will be available at:
http://localhost:8080/CalculatorWSApplication/CalculatorWS?WSDL

⚙️ **Client Application (Consumer Side)**

Now, we'll create a **Distributed Client Application** that consumes this web service.

✅ **Step 1: Create New Project**

- File → New Project → Java → Java Application
- Project Name: CalculatorClientApp

✅ **Step 2: Create Web Service Client**

- Right-click the project → **New** → **Web Service Client**
- In the dialog, enter the WSDL URL:
- http://localhost:8080/CalculatorWSApplication/CalculatorWS?WSDL
- Click **Finish.**

This automatically generates client proxy classes.

📄 **Source Code — MainClient.java**

```
package calculatorclientapp;

import org.me.calculator.CalculatorWS_Service;

public class MainClient {
```

```
    public static void main(String[] args) {
        try {
            // Create a service reference
            CalculatorWS_Service service = new CalculatorWS_Service();
            org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();

            // Invoke Web Service methods
            int a = 10, b = 5;

            System.out.println("Addition Result: " + port.add(a, b));
            System.out.println("Subtraction Result: " + port.subtract(a, b));
            System.out.println("Multiplication Result: " + port.multiply(a, b));
            System.out.println("Division Result: " + port.divide(a, b));
        } catch (Exception e) {
            System.err.println("Error occurred: " + e.getMessage());
        }
    }
}
```

✅ **Step 3: Run the Client**

Make sure your **CalculatorWSApplication** (server) is still running on GlassFish.
Then **Run the Client Project** → Output example:
Addition Result: 15
Subtraction Result: 5
Multiplication Result: 50
Division Result: 2.0

📜 **Summary:**

Component	Description
Server	CalculatorWS — A SOAP Web Service deployed on GlassFish
Client	Java Application consuming the service via WSDL
Tools	NetBeans IDE, GlassFish Server
Result	Successfully created, deployed, and consumed a web service

🏁 **Conclusion:**

Thus, a simple **Web Service** was created, deployed, and consumed by a **distributed Java client** application.
The experiment demonstrates how SOAP-based web services facilitate communication between distributed systems.