

## **ASSIGNMENT-5**

### **Aim and Objective:**

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

### **Tools/Environment:**

Java Programming Environment, JDK8, EclipseIDE with Tomcat Server

-

### **Calculator.java**

```
packagecom;
```

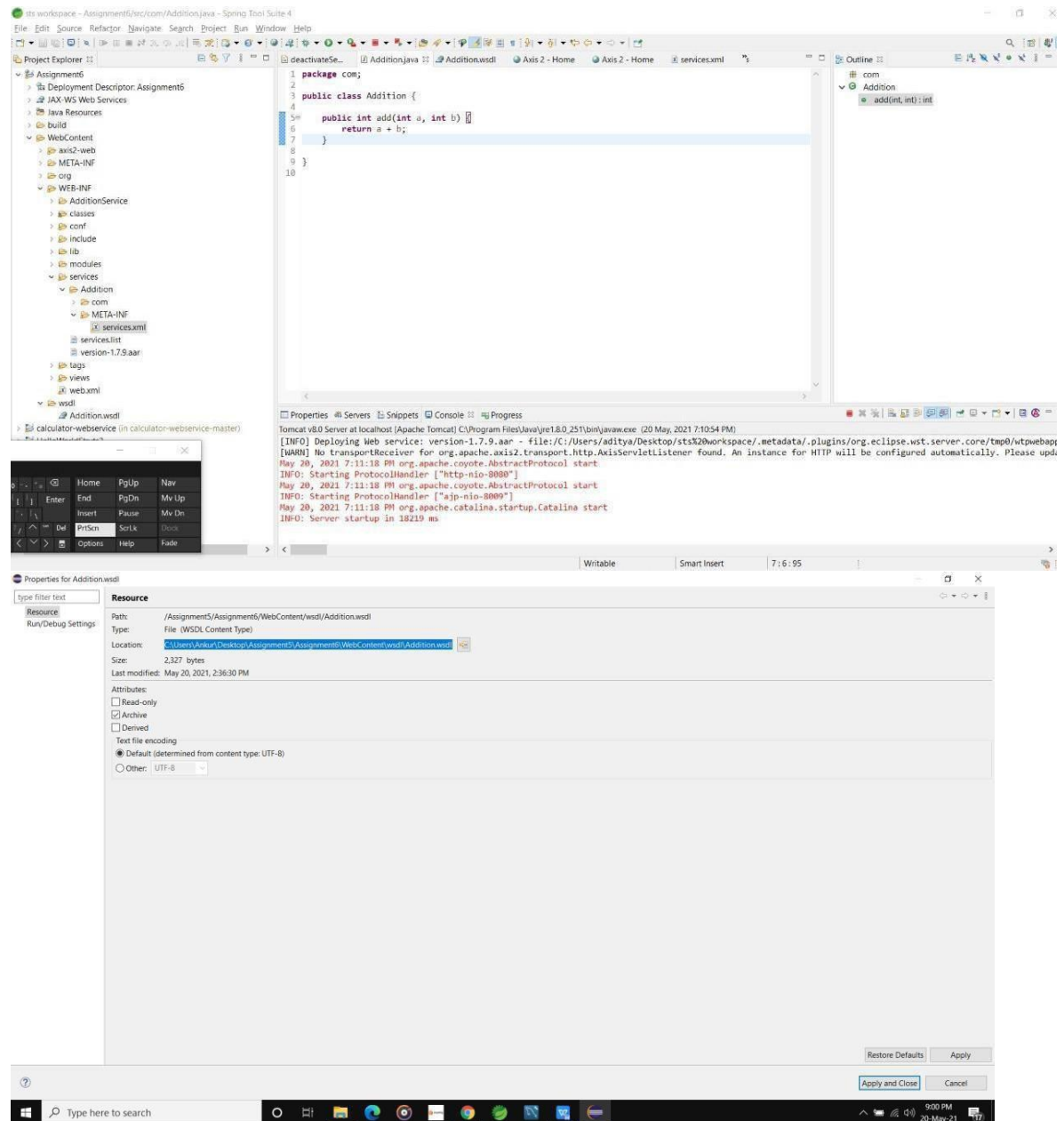
```
publicclassCalculator{  
    publicint add(inta,int b)  
    {  
        returna+b;  
    }  
    publicint sub(inta,int b)
```

```
    {  
        returna-b;  
    }  
    publicint mult(inta,int b)
```

```
    {  
        returna+b;  
    }  
    publicint div(inta,int b)
```

```
    {  
        returna+b;  
    }  
}
```

## Outputs



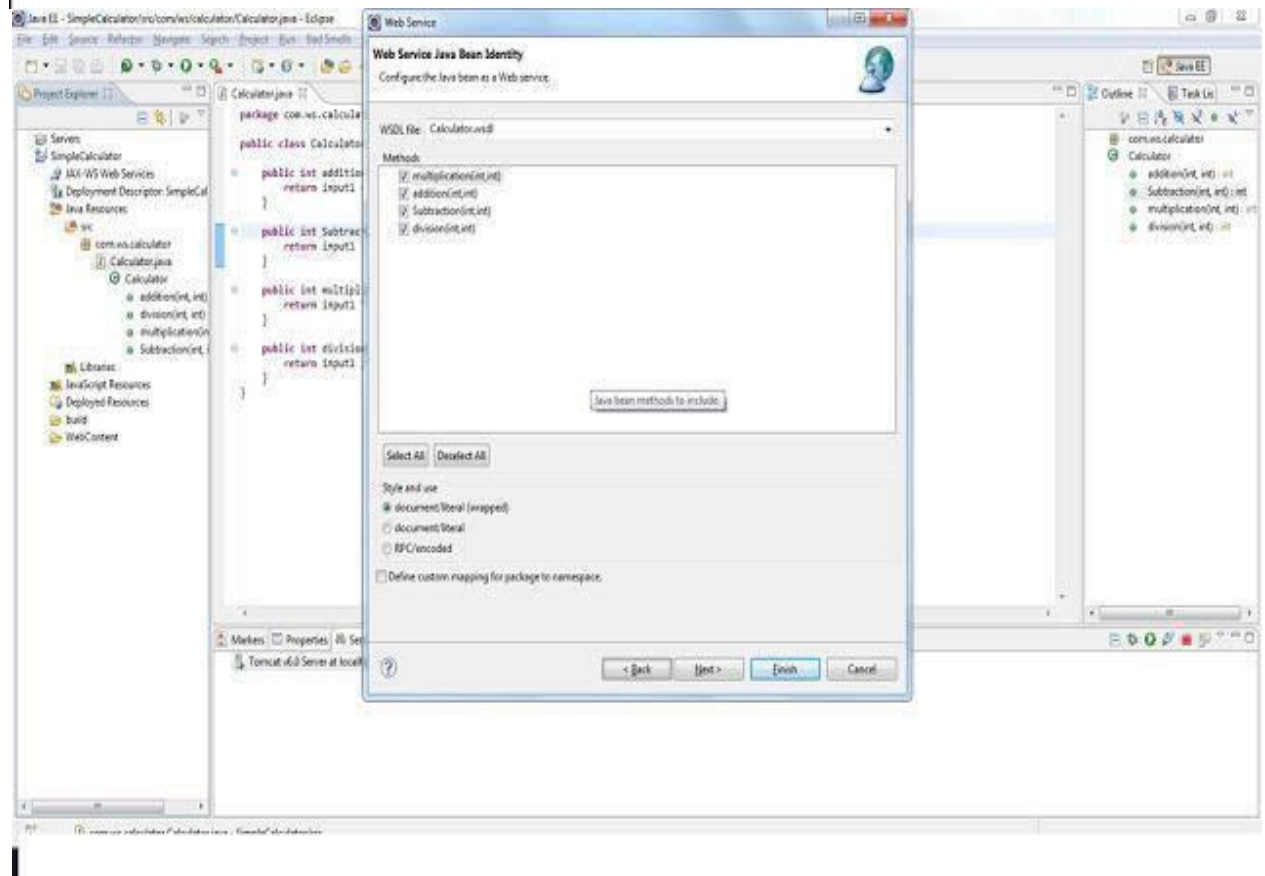
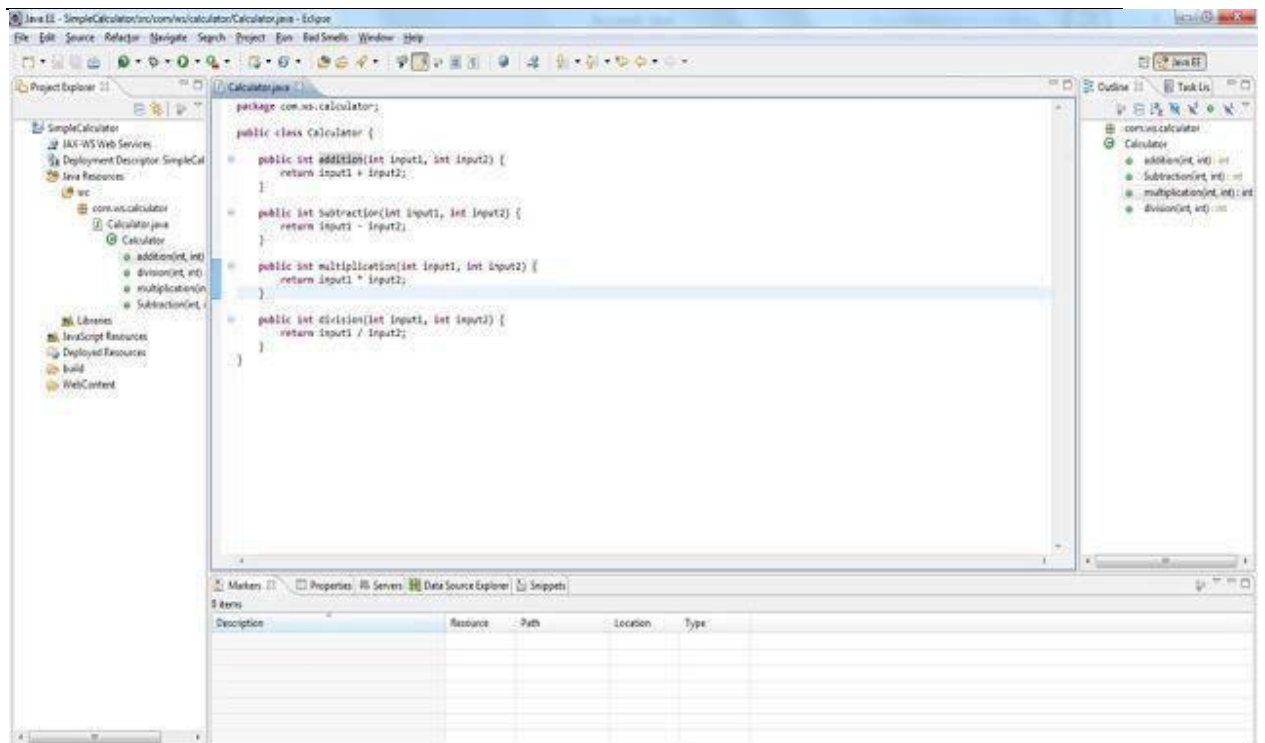
The image displays two software interfaces used for web service development and testing.

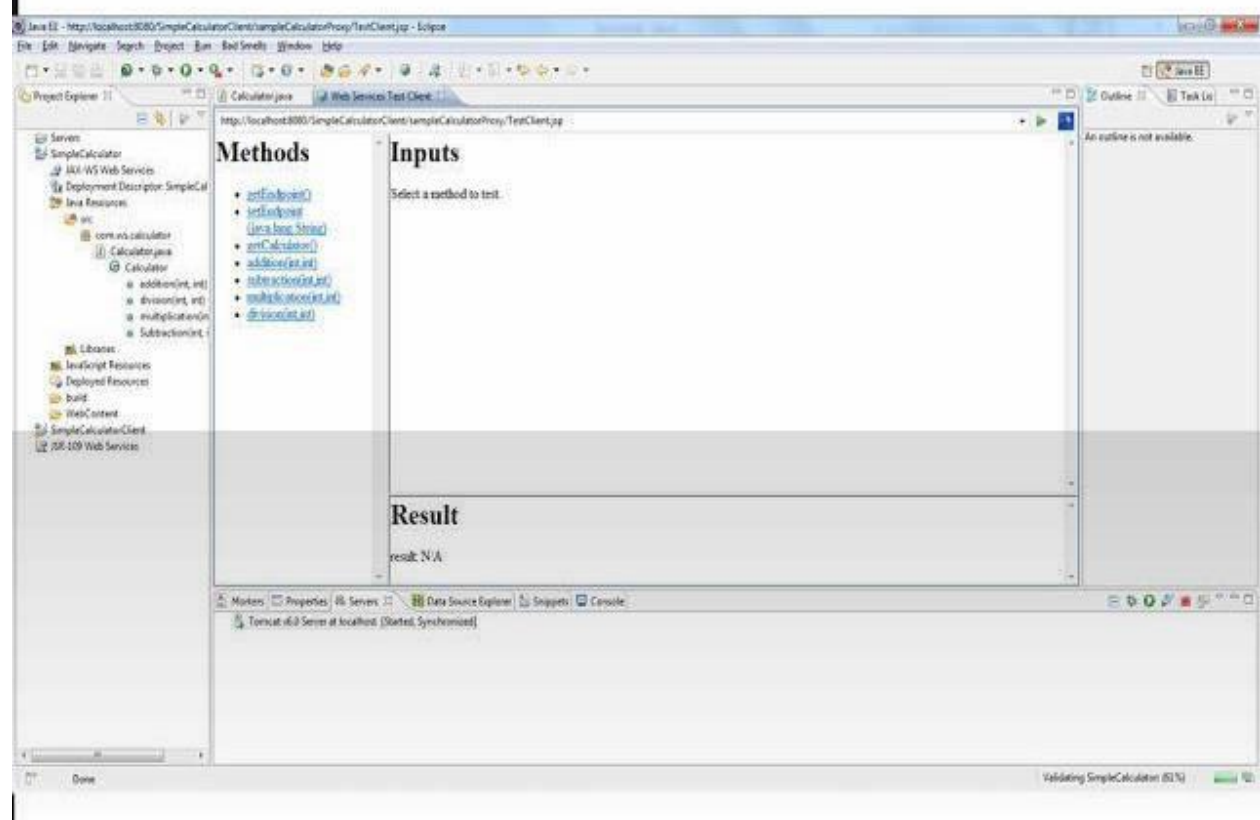
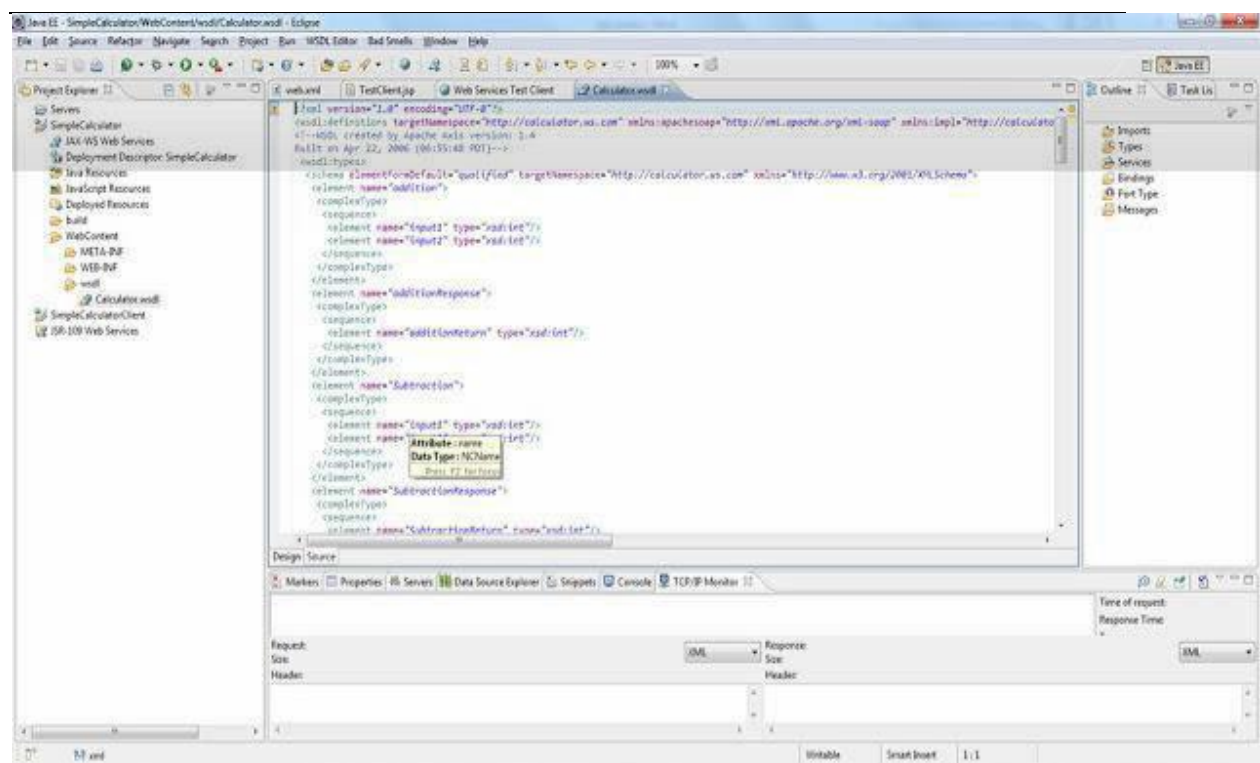
**Top Interface: SoapUI 5.6.0**

- Project Explorer:** Shows a project named "AdditionSoapBinding" with a sub-project "add" containing "Request 1".
- Main Window:** Displays the "Request 1" tab for the endpoint `http://localhost:8080/Assignment6/services/Addition`. It shows the SOAP request and response XML. The response includes a `com:a` element with value `1` and a `com:b` element with value `2`, resulting in a `com:retuen` element with value `3`.
- Bottom Panel:** Shows the "Headers (5)" tab with various headers like `Content-Type`, `Host`, and `SOAPAction`.

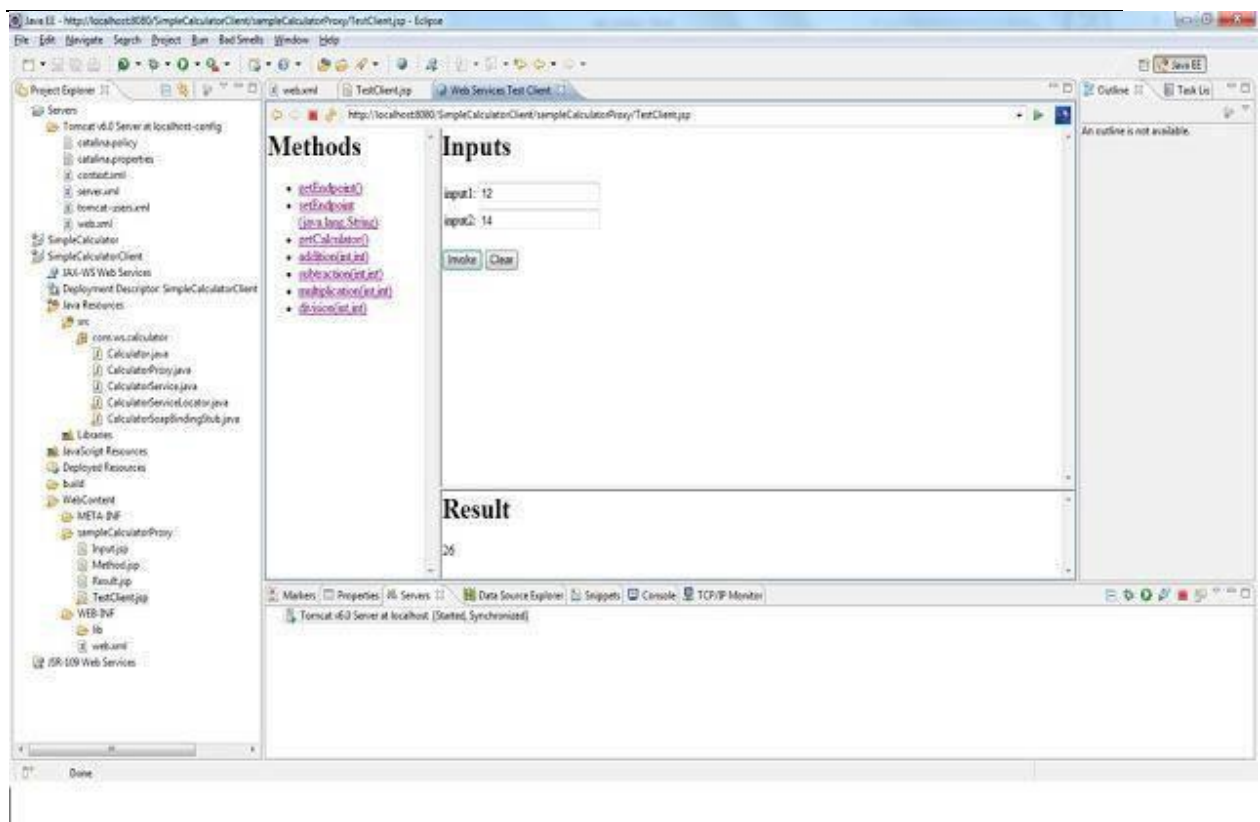
**Bottom Interface: Spring Tool Suite 4**

- Project Explorer:** Shows a project named "Assignment6" with a sub-project "services" containing "Addition" and "META-INF".
- Main Window:** Displays the "Web Services" configuration dialog for the "Addition" service. It shows the "Web service type" as "Bottom up Java bean Web Service" and the "Service implementation" as `com.Addition`. The "Client type" is set to "Java Proxy".
- Configuration:** The "Server runtime" is set to "Tomcat v8.0 Server" and the "Web service runtime" is set to "Apache Axis (Deprecated)".
- Buttons:** The "Finish" button is highlighted, indicating the configuration is complete.









```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:q0="http://calculator.ws.com" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns: xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <q0:addition>
      <q0:input1>1</q0:input1>
      <q0:input2>2</q0:input2>
    </q0:addition>
  </soapenv:Body>
</soapenv:Envelope>
```

1. Add the following code to index.jsp in index.jsp solid <%@page import="java.net.UR  
L, javax.xml.namespace.QName, javax.xml.ws.Service, org  
.apache.geronimo.samples.jws.Calculator"%> <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">  
<head> <title> Calculator </title> </head> <body> <form action="result.jsp"> Please enter 2 whole number to add: <i  
nput type="text" name="value1"> + <input type="text" name="value2"> <input type="submit" value="="> </form>  
</body> </html>
2. Right Click on the WebContent folder and Select **New**-> **JSP**.

3. Named the jsp as result.jsp and Select **Finish**.

Add the following code to jsp.

```
result.jsp<solid><%@pageimport "java.net.URL,javax.xml.namespace.QName,javax.xml.ws.Service,org.apache.geronimo.samples.jws.Calculator"%><htmlxmlns="http://www.w3.org/1999/xhtml"xml:lang="en"><head>
```

```
<title>Calculator
```

```
Result
```

```
</title>
```

```
</head>
```

```
<%intvalue1=0;intvalue2=0;intsum=0;try{System.out.println(request.getParameter("value1")+""+request.getParameter("value2"));value1=Integer.parseInt(request.getParameter("value1"));value2=Integer.parseInt(request.getParameter("value2"));URLurl=newURL("http://localhost:8080/jaxws-calculator-1.0/calculator?wsdl");QNameqname=newQName("http://jws.samples.geronimo.apache.org","Calculator");Service service=Service.create(url,qname);Calculator calc=(Calculator)service.getPort(Calculator.class);sum=calc.add(value1,value2);}catch(Exception e){e.printStackTrace();}%>
```

```
<body>The result is:<%=value1%> + <%=value2%> = <%=sum%> <br> <a href="index.jsp">Back</a> </body> </html> This finishes the development of Web client.
```

Expand WEB-INF/web.xml and add the following code web.xml solid

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

```
<web-app xmlns:calc="urn:geronimo-samples-jws" xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
```

```
<servlet>
```

```
<display-name>
```

```
name>CalculatorService
```

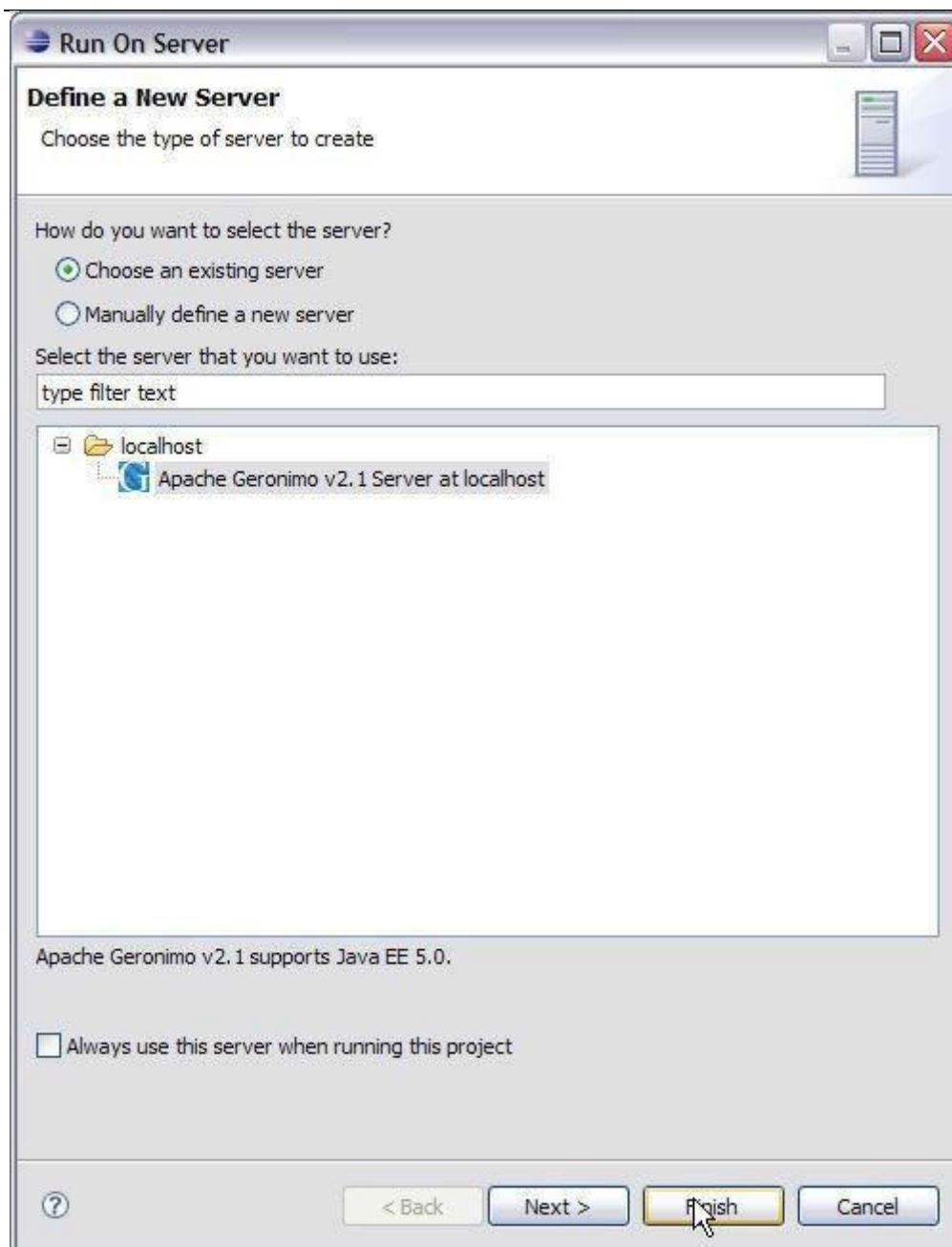
```
</display-name>
```

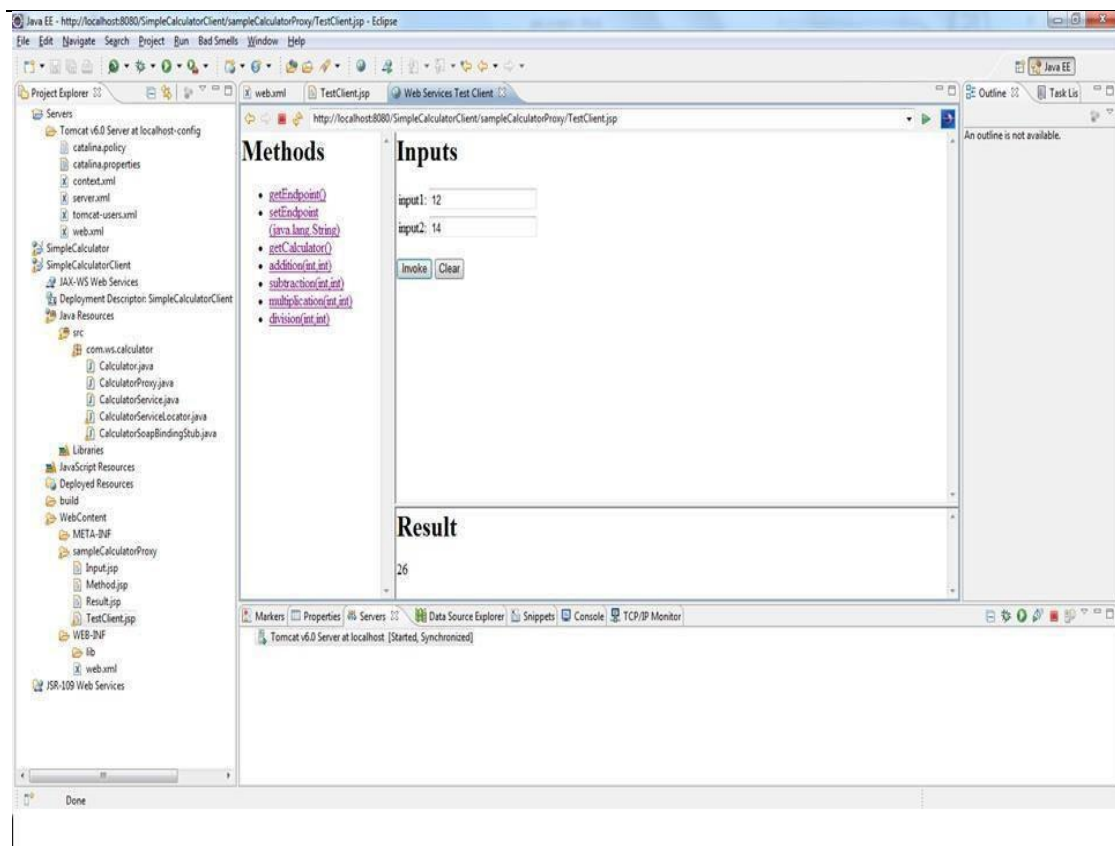
```
<servlet-  
name>CalculatorService  
</servlet-name>  
  
<servlet-  
class>org.apache.geronimo.samples.jws.CalculatorService  
</servlet-class>  
  
</servlet>  
  
<servlet-mapping>  
  
<servlet-  
name>CalculatorService  
</servlet-name>  
  
<url-pattern>  
  
/calculator  
</url-pattern>  
  
</servlet-mapping>  
  
<service-ref>  
  
<service-ref-  
name>services/Calculator</service-  
ref- name>  
  
<service-interface>javax.xml.ws.Service</service-interface>  
  
<wsdl-file>WEB-INF/CalculatorService.wsdl</wsdl-file> </service-ref>  
</web-app>
```

1. Similarly double click geronimo-web.xml and add the following code.  
geronimo-web.xml  

```
<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1">
```







```

package com.webServices;

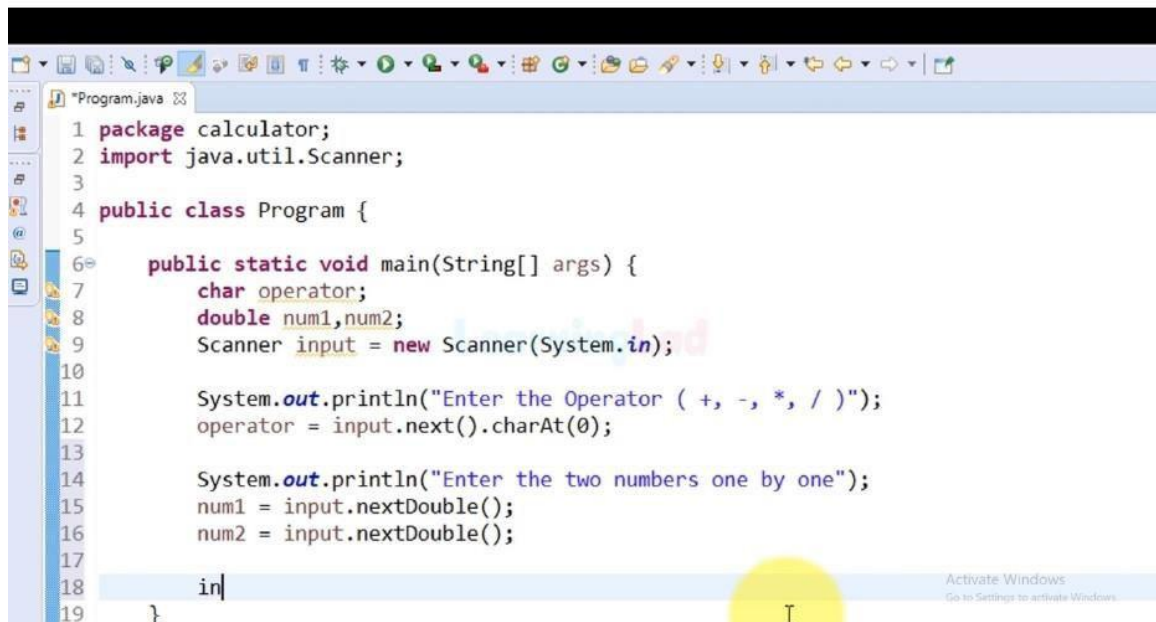
public class Calculator {
    public int add(int a, int b) { return (a+b); }

    public int subtract(int a, int b) { return (a-b); }

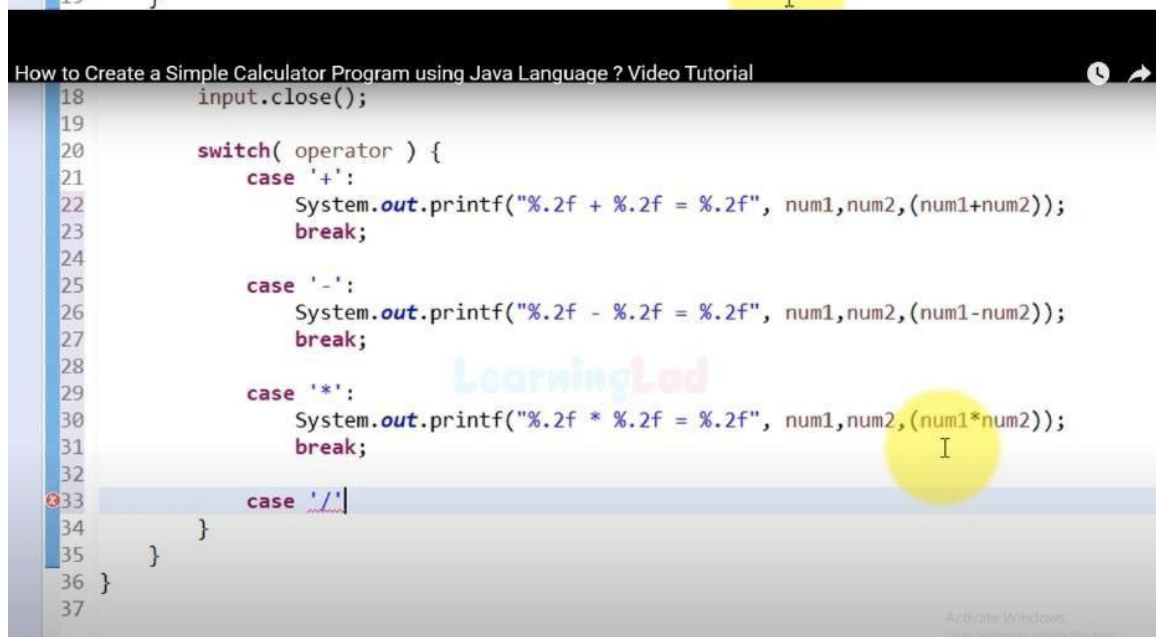
    public int multiply(int a, int b) { return (a*b); }

    public int divide(int a, int b) { return (a/b); }
}

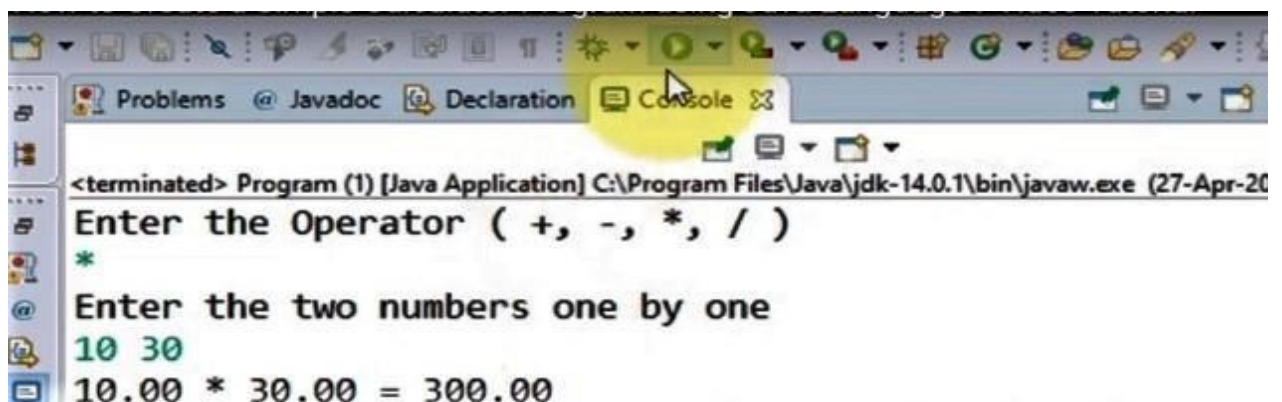
```



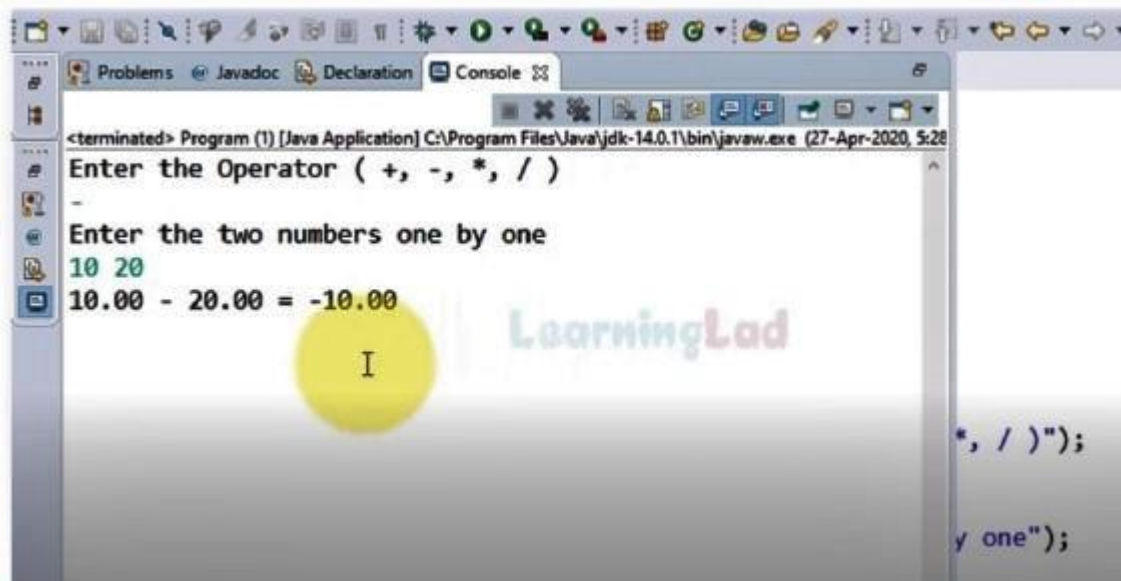
```
1 package calculator;
2 import java.util.Scanner;
3
4 public class Program {
5
6     public static void main(String[] args) {
7         char operator;
8         double num1,num2;
9         Scanner input = new Scanner(System.in);
10
11         System.out.println("Enter the Operator ( +, -, *, / )");
12         operator = input.next().charAt(0);
13
14         System.out.println("Enter the two numbers one by one");
15         num1 = input.nextDouble();
16         num2 = input.nextDouble();
17
18         in
19     }
```



```
18         input.close();
19
20         switch( operator ) {
21             case '+':
22                 System.out.printf("%.2f + %.2f = %.2f", num1,num2,(num1+num2));
23                 break;
24
25             case '-':
26                 System.out.printf("%.2f - %.2f = %.2f", num1,num2,(num1-num2));
27                 break;
28
29             case '*':
30                 System.out.printf("%.2f * %.2f = %.2f", num1,num2,(num1*num2));
31                 break;
32
33             case '/':
34
35         }
36     }
37 }
```

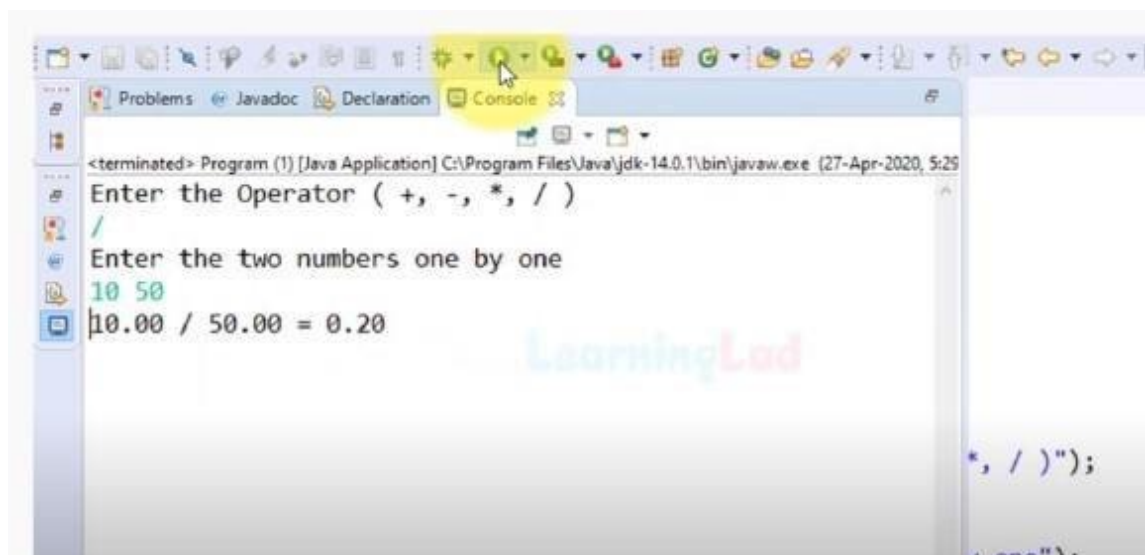


```
<terminated> Program (1) [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe (27-Apr-20
Enter the Operator ( +, -, *, / )
*
Enter the two numbers one by one
10 30
10.00 * 30.00 = 300.00
```



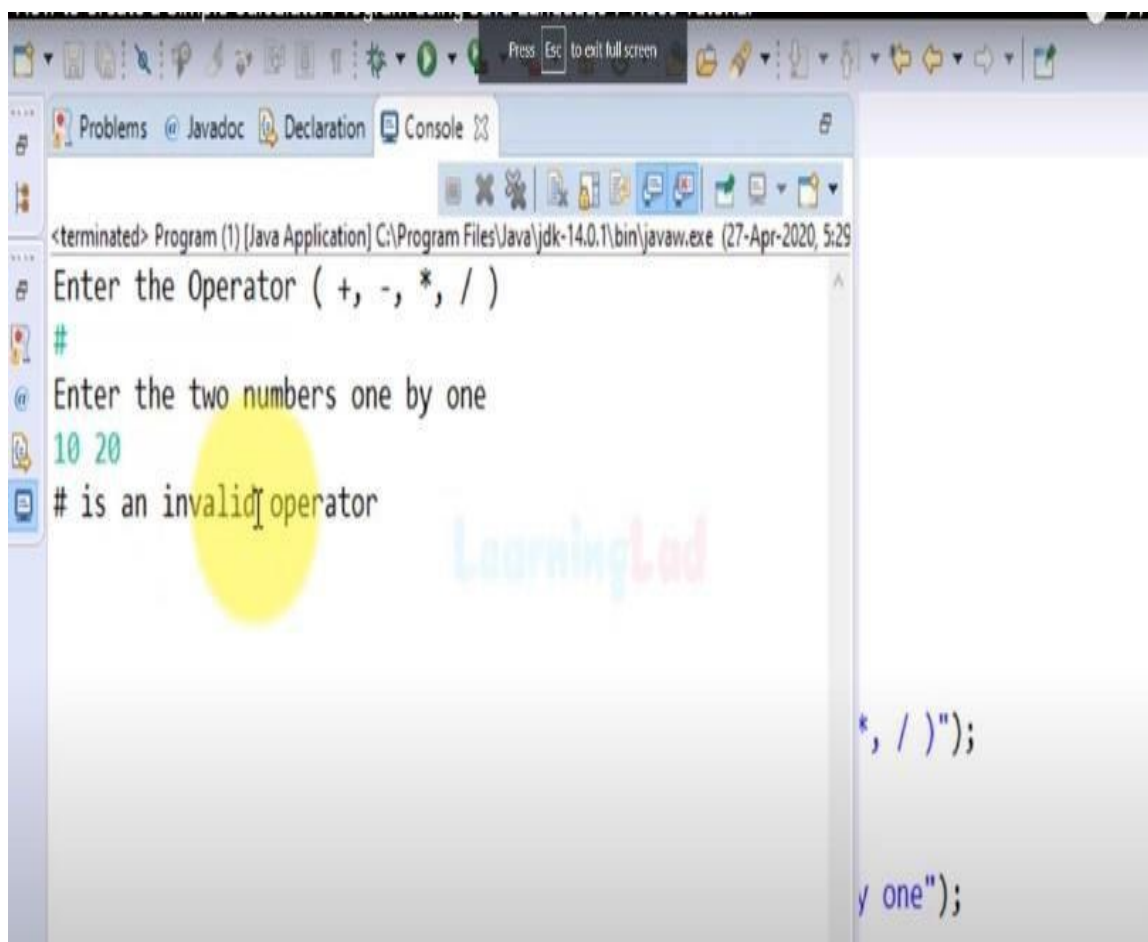
A screenshot of a Java IDE's console window. The title bar indicates the program is terminated. The console shows the following sequence of text: "Enter the Operator ( +, -, \*, / )", followed by a line break, then "Enter the two numbers one by one". The user has entered "10" and "20" on separate lines. The output displayed is "10.00 - 20.00 = -10.00". A yellow circle with the letter "I" is drawn over the output. A "LearningLad" watermark is visible in the background.

```
<terminated> Program (1) [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe (27-Apr-2020, 5:28)
Enter the Operator ( +, -, *, / )
-
Enter the two numbers one by one
10 20
10.00 - 20.00 = -10.00
```



A screenshot of the same Java IDE console window, but with the user input changed. The prompt "Enter the Operator ( +, -, \*, / )" is followed by a line break, then "Enter the two numbers one by one". The user has entered "10" and "50" on separate lines. The output displayed is "10.00 / 50.00 = 0.20". A yellow circle with the letter "I" is drawn over the output. A "LearningLad" watermark is visible in the background.

```
<terminated> Program (1) [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe (27-Apr-2020, 5:29)
Enter the Operator ( +, -, *, / )
/
Enter the two numbers one by one
10 50
10.00 / 50.00 = 0.20
```



```
<terminated> Program (1) [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe (27-Apr-2020, 5:29)
Enter the Operator ( +, -, *, / )
#
Enter the two numbers one by one
10 20
# is an invalid operator
```

The screenshot shows a Java IDE window with the 'Console' tab selected. The console output displays the execution of a Java program. The program prompts the user to 'Enter the Operator ( +, -, \*, / )'. The user has entered '#', which is highlighted in green. The program then prompts the user to 'Enter the two numbers one by one'. The user has entered '10 20', which is also highlighted in green. Finally, the program outputs an error message: '# is an invalid operator'. A yellow circle is drawn around the '#' character in the error message. The IDE interface includes a toolbar at the top with various icons and a status bar at the bottom.

## Theory:

### WebService:

A web service can be defined as a collection of open protocols and standards for exchanging information among systems or applications.

A service can be treated as a web service if:

- The service is discoverable through a simple lookup
- It uses a standard XML format for messaging
- It is available across internet/intranet networks.
- It is a self-describing service through a simple XML syntax
- The service is open to, and not tied to, any operating system/programming language

### Types of WebServices:

There are two types of web services:

1. **SOAP:** SOAP stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services. Since it's XML based, it's platform and language independent. So, our server can be based on JAVA and client can be on .NET, PHP etc. and vice-versa.
2. **REST:** REST(Representational State Transfer) is an architectural style for developing web services. It's getting popularity recently because it has small learning curve when compared to SOAP. Resources are core concepts of Restful web services and they are uniquely identified by their URIs.

### Web service architectures:

As part of a web service architecture, there exist three major roles.

**Service Provider** is the program that implements the service agreed for the web service and exposes the service over the internet/intranet for other applications to interact with.

**Service Requestor** is the program that interacts with the webservice exposed by the Service Provider. It

Makes an invocation to the web service over the network to the Service Provider and exchanges information.

**Service Registry** acts as the directory to store references to the web services



---

The following are the steps involved in a basic SOAP web service operational behavior:

1. The client program that wants to interact with another application prepares its request content as a SOAP message.
2. Then, the client programs ends this SOAP message to the server web service as an HTTP POST request with the content passed as the body of the request.
3. The web service plays a crucial role in this step by understanding the SOAP request and converting it into a set of instructions that the server program can understand.
4. The server program processes the request content as programmed and prepares the output as the response to the SOAP request.
5. Then, the web service takes this response content as a SOAP message and reverts to the SOAP HTTP request invoked by the client program with this response.
6. The client program web service reads the SOAP response message to receive the outcome of the server program for the request content it sent as a request.

### **SOAP web services:**

**Simple Object Access Protocol(SOAP)** is an XML- based protocol for accessing web services. It is a W3C recommendation for communication between two applications, and it is a platform-and language- independent technology in integrated distributed applications.

While XML and HTTP together make the basic platform for webservices, the following are the key components of standard SOAP web services:

**Universal Description, Discovery, and Integration(UDDI):**UDDI is an XML based framework for describing, discovering ,and integrating web services. It acts as a directory of web service interfaces described in the WSDL language.

### **Web Services Description Language(WSDL):**

WSDL is an XML document containing information about web services, such as the method name, method parameters, and how to invoke the service. WSDL is part of the UDDI registry. It acts as an interface between applications that want to interact based on web services.

### **Conclusion:**

This assignment, described the Web services approach to the Service Oriented Architecture concept. Also, described the Java APIs for programming Web services and demonstrated examples of their use by providing detailed step-by- step examples of how to program Web services in Java.