LABORATORY RECORD

Name : **Monica Rachel Rani V**

Roll No : **2016242012**

Class : **M.Sc (INTEGRATED COURSE)**

Course code & Title : XT7961 & SERVICE ORIENTED ARCHITECTURE

DEPARTMENT OF MATHEMATICS

COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY

CHENNAI - 600 025

DEPARTMENT OF MATHEMATICS
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI - 600 025

BONAFIDE CERTIFICATE

Name : Monica Rachel Rani V

Roll No : 2016242012 **Programme : M.Sc(Integrated)**

Branch : Information Technology **Semester : 9**

Certified to be the bonafide record of work done by the above student in **XT7961 – Service Oriented Architecture** *Laboratory course during the Semester* **IX** *of the academic year* **2020-21** *submitted for the Practical Examination held on* **20/10/2020**

Lab Course Instructor **Head of the Department**

INDEX

Ex. No: 1
Date: 14/09/20

SOAP Implementation for a Calculator

AIM:

To implement client and server using SOAP Architecture for a Calculator

PROCEDURE:

Server:

1. Open NetBeans
2. Go to New > Project > Java Web > Web Application
3. Enter Project name and finish the creation process
4. Right click on the Project, go to New > Web Service
5. Provide Service and Package name and finish the creation process.
6. In the .java file, Go to Design View and click on "Add Operation"
7. Provide a name for the Operation and choose the number of parameters needed along with the data type, check mark the final and give OK
8. In the Source View type the statements required to do the operation and define the return type
9. Right click on the Project and choose clean and build
10. Again right click on the Project and choose deploy
11. Right click on the Web Services name and choose Test Web Services

Client:

1. Open NetBeans
2. Go to New > Project > Java Web > Web Application
3. Enter Project name and finish the creation process
4. Right click on the Project, go to New > Web Service Client
5. Browse the Project name, and choose the server's web service name and choose Finish
6. Right click on the Web Pages and create new JSP file
7. In the JSP file, create a form to fetch the user input
8. Right click on the form in the JSP file and choose Web Services Client Resource > Call Web Service Operation and choose the required operations to be performed.
9. In the try block, type the required code to fetch the data from the Web Page to the function invoking the operation, and to print the result in the Web Page
10. Run the JSP file

CODE:

Server:

```
package SOA;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
@WebService(serviceName = "Server")
public class Server {
   @WebMethod(operationName = "Add")
   public int Add(@WebParam(name = "a") final int a, @WebParam(name = "b") final int b) {
      return a+b;
   }
```

```
    @WebMethod(operationName = "Sub")
    public int Sub(@WebParam(name = "a") final int a, @WebParam(name = "b") final int b) {
      if (a<b){
         return b-a;
      }
      else
       return a-b;
    }
    @WebMethod(operationName = "Mul")
    public int Mul(@WebParam(name = "a") final int a, @WebParam(name = "b") final int b) {
      return a*b;
    }
    @WebMethod(operationName = "Div")
    public String Div(@WebParam(name = "a") final int a, @WebParam(name = "b") final int b) {
      float c;
       String s;
      if (b==0){
         return "B value should not be equal to 0";
      }
      else
         c = (a/b);
         s = Float.toString(c);
         return s;
    }}
```

Client:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <form action="newjsp.jsp">
            <p> Addition </p>
            <input type="text" name="a" placeholder="add1"><br><br>
            <input type="text" name="b" placeholder="add2"><br><br>
            <p> Subtraction </p>
            <input type="text" name="c" placeholder="sub1"><br><br>
            <input type="text" name="d" placeholder="sub2"><br><br>
            <p> Multiplication </p>
            <input type="text" name="e" placeholder="mul1"><br><br>
            <input type="text" name="f" placeholder="mul2"><br><br>
            <p> Division </p>
            <input type="text" name="g" placeholder="div1"><br><br>
            <input type="text" name="h" placeholder="div2"><br><br>
            <input type="submit" value="Calculate">
        </form>
    <%-- start web service invocation --%><hr/>
    <%
    try {
```

```
        soa.Server_Service service = new soa.Server_Service();
        soa.Server port = service.getServerPort();
        String a = request.getParameter("a");
        String b = request.getParameter("b");
         String c = request.getParameter("c");
        String d = request.getParameter("d");
        String e = request.getParameter("e");
        String f = request.getParameter("f");
        String g = request.getParameter("g");
        String h = request.getParameter("h");
        int aa= Integer.parseInt(a);
        int bb= Integer.parseInt(b);
        int cc= Integer.parseInt(c);
        int dd= Integer.parseInt(d);
        int ee= Integer.parseInt(e);
        int ff= Integer.parseInt(f);
        int gg= Integer.parseInt(g);
        int hh= Integer.parseInt(h);
        int addre = port.add(aa, bb);
        out.println("Addition = "+addre);
        int subre = port.sub(cc, dd);
        out.println("Subtraction = "+subre);
         int mulre = port.mul(ee, ff);
        out.println("Multiplication = "+mulre);
        String s = port.div(gg, hh);
        out.println("Division = "+s);
    } catch (Exception ex) { }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

OUTPUT:

Server:

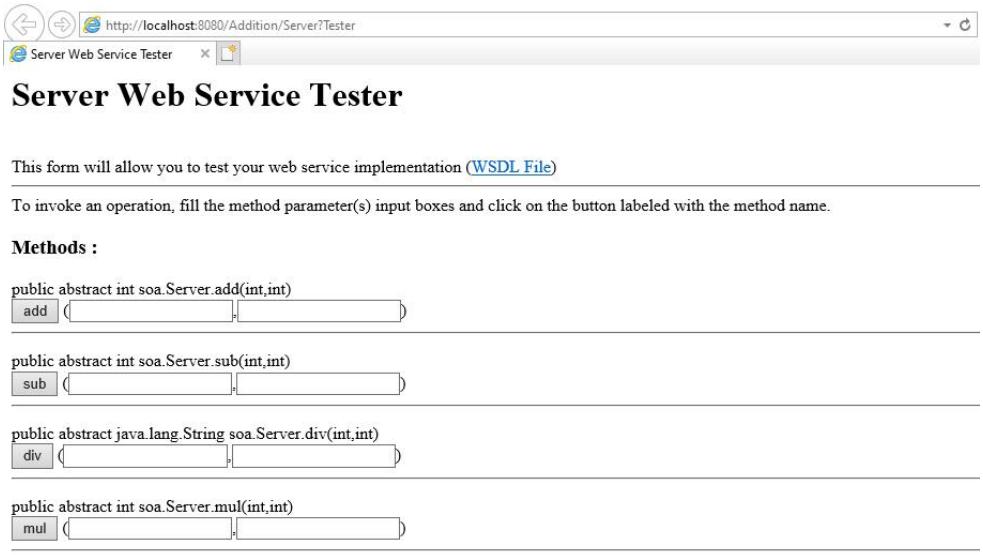


Fig 1.1 Web Service Index Page

Fig 1.2 Division Invocation Page

Client:

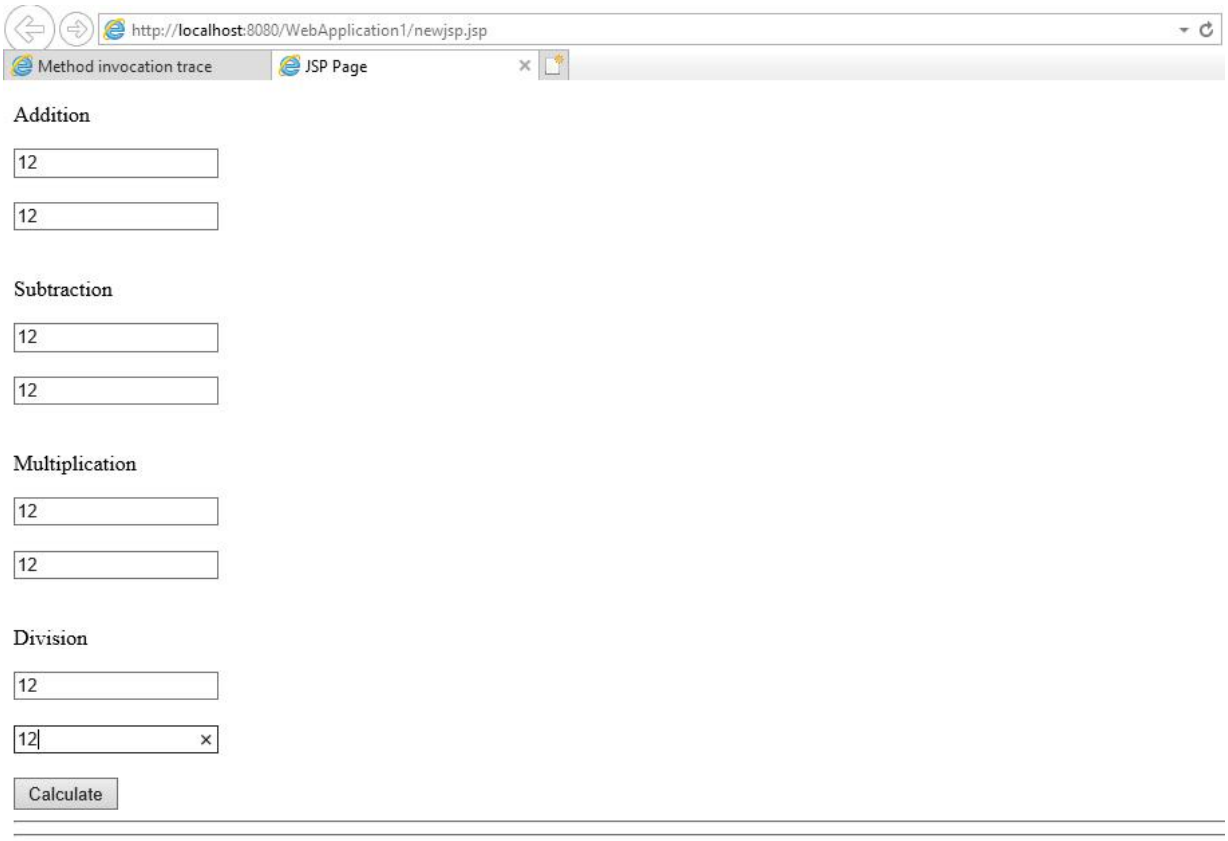


Fig 1.3 Client JSP Page

Fig 1.4 Client JSP Page with Output

RESULT:

Thus, the client and server was implemented using the SOAP Architecture.

Ex. No: 2
Date: 08/09/20

Web Services using ASP.NET – Temperature Conversion

AIM:

To implement Web Services using ASP.NET

PROCEDURE:

ASP.NET Web Application(Server):

1. Open Microsoft Visual Studio 2019
2. Go to New -> Project and select ASP.NET Web Application(.NET Framework)
3. Enter name, Select Empty Template and finish the creation process.
4. Right click on the Project and choose Add-> New Item and select WebService (ASMX) file
5. Enter name and finish the creation process
6. In the .asmx.cs file, create seperate Web Method for the converting Celsius to Fahrenheit and Fahrenheit to Celsius
7. Run the Project
8. Click on the links with the function name to execute the functions by providing the inputs required
9. Click on the link provided as Service Description, copy the redirected Web Page's URL which is in WSDL format
10. Right click on the project and choose Add-> Service Reference
11. Add the URL copied in the Address field and give Go, which lists the Web service name and the operations
12. Enter name and finish the creation process

Web Form(Client):

13. Right click on the Project and choose Add-> New Item and select Web Form
14. Enter name the finish the creation process
15. In the .aspx file, switch to design mode and design the form to fetch data and output data with appropriate textboxes, labels and buttons
16. In the .aspx.cs file pass the fetched data to the functions in the Web Service
17. Run the Project

CODE:

WebService1.asmx.cs (Server):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
namespace final
{
   [WebService(Namespace = "http://tempuri.org/")]
   [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
   [System.ComponentModel.ToolboxItem(false)]
   public class WebService1 : System.Web.Services.WebService
   {
```

```
[WebMethod]
        public float CtoF(float a)
        {
            float f = ((a * 9) / 5) + 32;
            return f;
        }

[WebMethod]
        public float FtoC(float a)
        {
            float c= (a - 32) * 5 / 9;
            return c;
        }
    }
}
```

WebForm1.aspx (Client):

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="final.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <style type="text/css">
        #form1 {
            height: 194px;
        }
    </style>
</head>
<body>
 <asp:Label ID="Label3" runat="server" Text="Enter Celsius"  style="margin-left: 22px" ></asp:Label>
        <asp:TextBox ID="TextBox5" runat="server" style="margin-left: 53px"
Width="96px"></asp:TextBox>
        <asp:Button ID="Button3" runat="server" OnClick="Button3_Click" style="margin-left: 72px"
Text="To Fahrenheit" Width="107px" />
        <asp:TextBox ID="TextBox6" runat="server" style="margin-left: 66px"></asp:TextBox><br><br>
        <asp:Label ID="Label4" runat="server" Text="Enter Fahrenheit"  style="margin-left: 22px"
></asp:Label>
        <asp:TextBox ID="TextBox7" runat="server" style="margin-left: 33px"
Width="96px"></asp:TextBox>
        <asp:Button ID="Button4" runat="server" OnClick="Button4_Click" style="margin-left: 71px"
Text="To Celsius" Width="107px" />
        <asp:TextBox ID="TextBox8" runat="server" style="margin-left: 66px"></asp:TextBox><br><br>
    </form>
</body>
</html>
```

WebForm1.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Security.Cryptography;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace final
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {

      }
protected void Button3_Click(object sender, EventArgs e)
      {
         WebService1 ws = new WebService1();
         String a = TextBox5.Text;
         float a1 = float.Parse(a);
         float val = ws.CtoF(a1);
         TextBox6.Text = val.ToString();
      }
      protected void Button4_Click(object sender, EventArgs e)
      {
         WebService1 ws = new WebService1();
         String a = TextBox7.Text;
         float a1 = float.Parse(a);
         float val = ws.FtoC(a1);
         TextBox8.Text = val.ToString();
      }
   }
}
```

OUTPUT:

Server:

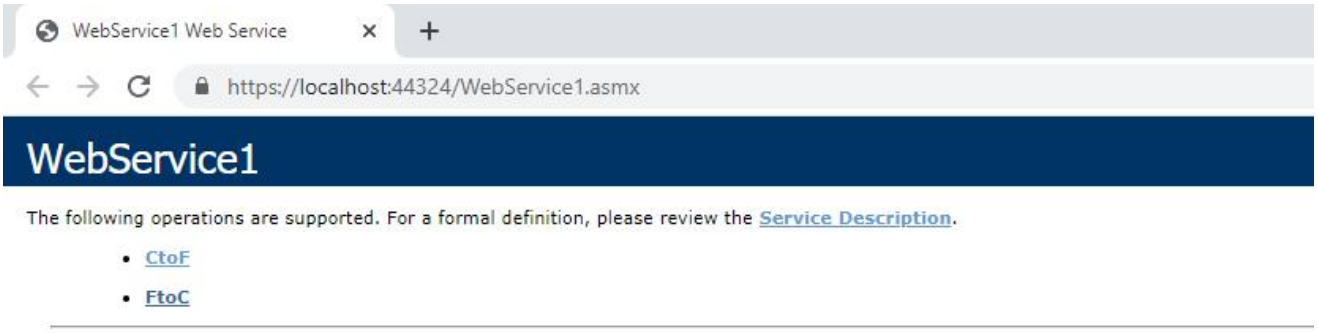


Fig 2.1 Web Service Invocation

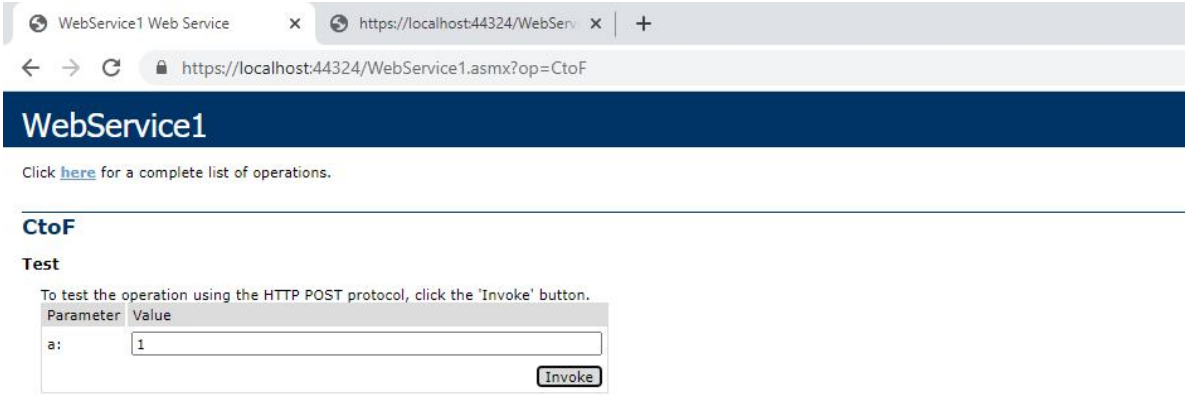


Fig 2.2 Invocation of Celsius to Fahrenheit Function

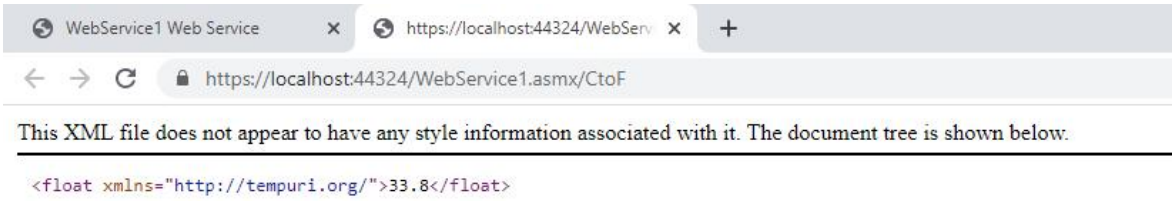


Fig 2.3 Output of Celsius to Fahrenheit Function

Client:

Fig 2.4 Web Form

Fig 2.5 Web Form Output

RESULT:

Thus, Web Services were used to convert Celsius to Fahrenheit and Fahrenheit to Celsius using C#.

Ex. No: 3
Date: 15/09/20

Web Services using ASP.NET – Finding Permutation, Combination, Factorial and Fibonacci Series

AIM:

To implement Web Services using ASP.NET for finding Permutation, Combination, Factorial and Fibonacci Series

PROCEDURE:

ASP.NET Web Application(Server):

1. Open Microsoft Visual Studio 2019
2. Go to New -> Project and select ASP.NET Web Application(.NET Framework)
3. Enter name, Select Empty Template and finish the creation process.
4. Right click on the Project and choose Add-> New Item and select WebService (ASMX) file
5. Enter name and finish the creation process
6. In the .asmx.cs file, create seperate Web Method for the finding Permutation, Combination, Factorial and Fibonacci Series
7. Run the Project
8. Click on the links with the function name to execute the functions by providing the inputs required
9. Click on the link provided as Service Description, copy the redirected Web Page's URL which is in WSDL format
10. Right click on the project and choose Add-> Service Reference
11. Add the URL copied in the Address field and give Go, which lists the Web service name and the operations
12. Enter name and finish the creation process

Web Form(Client):

1. Right click on the Project and choose Add-> New Item and select Web Form
2. Enter name the finish the creation process
3. In the .aspx file, switch to design mode and design the form to fetch data and output data with appropriate textboxes, labels and buttons
4. In the .aspx.cs file pass the fetched data to the functions in the Web Service
5. Run the Project

CODE:

WebService1.asmx.cs (Server):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

namespace final
{
   [WebService(Namespace = "http://tempuri.org/")]
   [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
```

```
[System.ComponentModel.ToolboxItem(false)]
public class WebService1 : System.Web.Services.WebService
{

    [WebMethod]
    public int Factorial(int a)
    {
        if (a == 0)
            return 1;
        return a * Factorial(a - 1);
    }

    [WebMethod]
    public int Fibonacci(int a)
    {
        int firstnumber = 0, secondnumber = 1, result = 0;

        if (a == 0) return 0;
        if (a == 1) return 1;


        for (int i = 2; i <= a; i++)
        {
            result = firstnumber + secondnumber;
            firstnumber = secondnumber;
            secondnumber = result;
        }

        return result;
    }
    [WebMethod]
    public String Percom(int a, int b) {
      int  per = factorial(a) / factorial(a - b);
        int comb = factorial(a) / (factorial(b) * factorial(a - b));
        int factorial(int n)
        {
            int fact = 1;
            int i = 1;
            while (i <= n)
            {
                fact *= i;
                i++;
            }
            return fact;
        }
        String s = "Permuation = " + per +" " + "Combination = " + comb;
        return s;
    }
}
}
```

WebForm1.aspx (Client):

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="final.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title></title>
   <style type="text/css">
      #form1 {
         height: 194px;
      }
   </style>
</head>
<body>
   <form id="form1" runat="server">
      <div>
      </div>

     <asp:Label ID="Label1" runat="server" Text="Enter Value"  style="margin-left: 22px"></asp:Label>
      <asp:TextBox ID="one" runat="server" style="margin-left: 62px" Width="96px"></asp:TextBox>
      <asp:Button ID="Button2" runat="server" OnClick="Button2_Click" style="margin-left: 71px"
Text="Factorial" Width="107px" />
      <asp:TextBox ID="TextBox4" runat="server" style="margin-left: 68px"></asp:TextBox>
      <br><br>

      <asp:Label ID="Label2" runat="server" Text="Enter Value" style="margin-left: 22px"></asp:Label>
      <asp:TextBox ID="two" runat="server" style="margin-left: 61px" Width="96px"></asp:TextBox>
      <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Fibonacci"
style="margin-left: 72px" Width="104px" />
      <asp:TextBox ID="TextBox3" runat="server" style="margin-left: 70px"
Width="172px"></asp:TextBox>
      <br><br>

     <asp:Label ID="Label3" runat="server" Text="Enter N"  style="margin-left: 22px" ></asp:Label>
      <asp:TextBox ID="TextBox5" runat="server" style="margin-left: 83px"
Width="95px"></asp:TextBox><br><br>
       <asp:Label ID="Label4" runat="server" Text="Enter R"  style="margin-left: 22px" ></asp:Label>
      <asp:TextBox ID="TextBox7" runat="server" style="margin-left: 86px"
Width="91px"></asp:TextBox>
      <asp:Button ID="Button3" runat="server" OnClick="Button3_Click" style="margin-left: 40px"
Text="Permutation and Combination" Width="194px" />
      <asp:TextBox ID="TextBox6" runat="server" style="margin-left: 18px"
Width="234px"></asp:TextBox>
      <br><br>
   </form>
</body>
</html>
```

WebForm1.aspx.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace final
{
   public partial class WebForm1 : System.Web.UI.Page
   {
      protected void Page_Load(object sender, EventArgs e)
      {

      }

      protected void Button1_Click(object sender, EventArgs e)
      {
         WebService1 ws = new WebService1();
         String b = two.Text;
         int b1 = Int32.Parse(b);
         int val = ws.Fibonacci(b1);
         TextBox3.Text = val.ToString();

      }

      protected void Button2_Click(object sender, EventArgs e)
      {
         WebService1 ws = new WebService1();
         String a = one.Text;
         int a1 = Int32.Parse(a);
         int c1 = ws.Factorial(a1);
         TextBox4.Text = c1.ToString();
      }

      protected void Button3_Click(object sender, EventArgs e)
      {
         WebService1 ws = new WebService1();
         String a = TextBox5.Text;
         String b = TextBox7.Text;
         int a1 = Int32.Parse(a);
         int b1 = Int32.Parse(b);
         String val = ws.Percom(a1,b1);
         TextBox6.Text = val;
      }
}
}
```

OUTPUT:

Server:

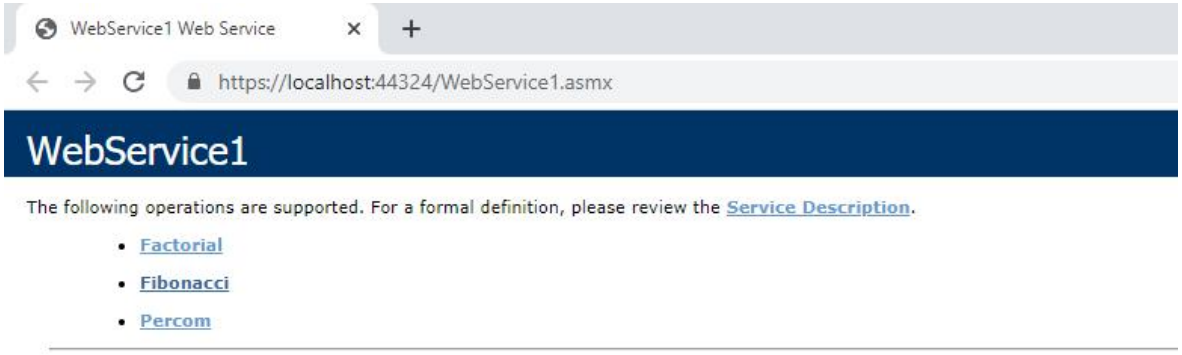


Fig 3.1 Web Service Invocation

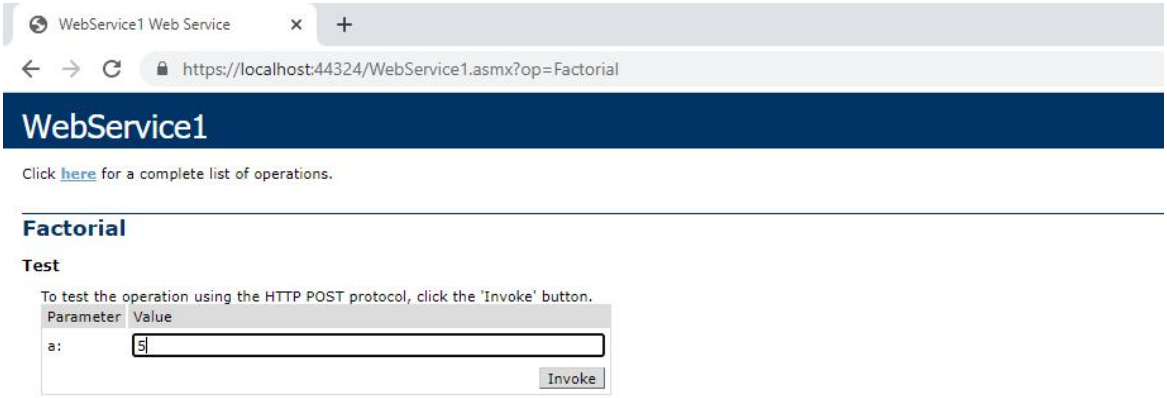


Fig 3.2 Invocation of Factorial Function

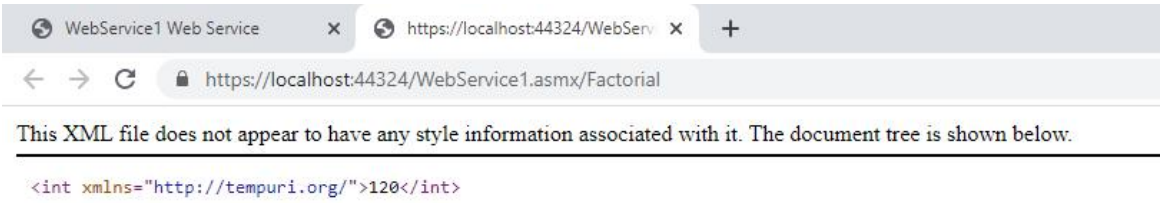


Fig 3.3 Output of Factorial Function

Client:

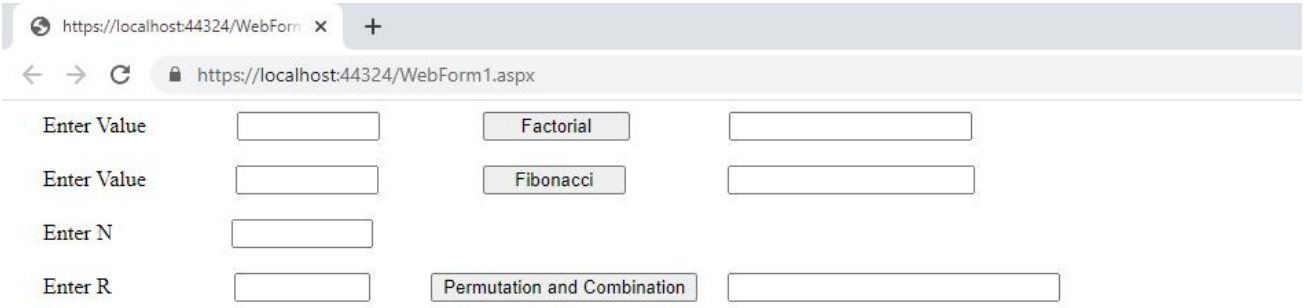
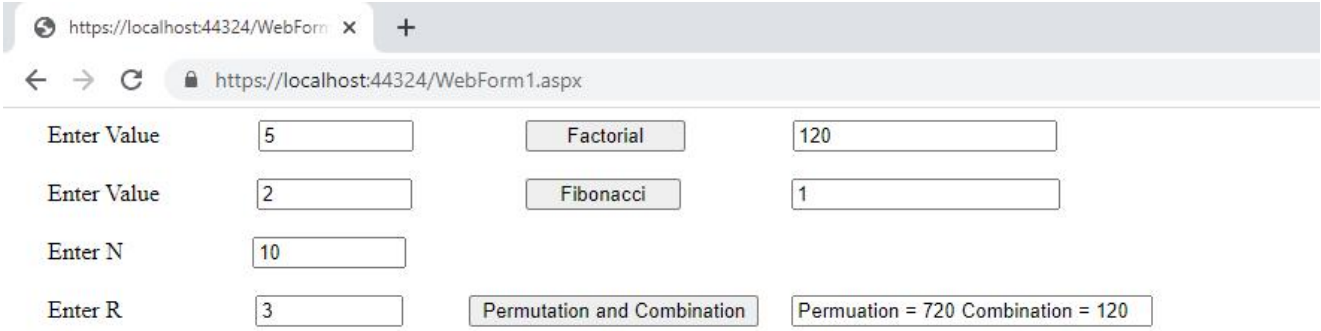


Fig 3.4 Web Form



3.5 Web Form Output

RESULT:

Thus, Web Services were used to compute Fibonacci, Factorial, Permutation and Combination, using C#.

Ex. No: 4
Date: 22/09/20

Web Services using Java - Simple and Compound Interest Calculation

AIM:

To find Simple and Compound Interest using Web Services in JAVA

PROCEDURE:

Server:

1. Open NetBeans
2. Go to New > Project > Java Web > Web Application
3. Enter Project name and finish the creation process
4. Right click on the Project, go to New > Web Service
5. Provide Service and Package name and finish the creation process.
6. In the .java file, Go to Design View and click on "Add Operation"
7. Provide a name for the Operation and choose the number of parameters needed along with the data type, check mark the final and give OK
8. In the Source View type the statements required to do the operation and define the return type
9. Right click on the Project and choose clean and build
10. Again right click on the Project and choose deploy
11. Right click on the Web Services name and choose Test Web Services

Client:

1. Open NetBeans
2. Go to New > Project > Java Web > Web Application
3. Enter Project name and finish the creation process
4. Right click on the Project, go to New > Web Service Client
5. Browse the Project name, and choose the server's web service name and choose Finish
6. Right click on the Web Pages and create new JSP file
7. In the JSP file, create a form to fetch the user input
8. Right click on the form in the JSP file and choose Web Services Client Resource > Call Web Service Operation and choose the required operations to be performed.
9. In the try block, type the required code to fetch the data from the Web Page to the function invoking the operation, and to print the result in the Web Page
10. Run the JSP file

CODE:

Server:

```
package pack;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;
@WebService(serviceName = "NewWebService")
public class NewWebService {
@WebMethod(operationName = "SInterest")
public float Interest(@WebParam(name = "p") final float p, @WebParam(name = "n") final float n,
@WebParam(name = "r") final float r) {
    float si = (p*n*r)/100;
```

```
        return si; }
@WebMethod(operationName = "CInterest")
public float CInterest(@WebParam(name = "p") final float p, @WebParam(name = "n") final float n,
@WebParam(name = "r") final float r, @WebParam(name = "t") final float t) {
        float ci = (float) (p * Math.pow(1 + (r / n), n * t));
        return ci;
}}
```

Client:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body><form action="index.jsp">
            <h1>Simple Interest</h1>
            <input type="text" name="sp" placeholder="Principal"><br>
            <input type="text" name="sn" placeholder="Number of times/Year"><br>
            <input type="text" name="sr" placeholder="Rate"><br>
            <h1>Compound Interest</h1>
            <input type="text" name="cp" placeholder="Principal"><br>
            <input type="text" name="cn" placeholder="Number of times/Year"><br>
            <input type="text" name="cr" placeholder="Rate"><br>
            <input type="text" name="ct" placeholder="time"><br>
            <input type="submit" value="Calculate">
        </form>
    <%-- start web service invocation --%><hr/>
    <%
    try {
            pack.NewWebService_Service service = new pack.NewWebService_Service();
            pack.NewWebService port = service.getNewWebServicePort();
                    String p = request.getParameter("sp");
      String n = request.getParameter("sn");
      String r = request.getParameter("sr");
                    float pp = Float.parseFloat(p);
                    float nn = Float.parseFloat(n);
      float rr = Float.parseFloat(r);
                    float result = port.sInterest(pp, nn, rr);
            out.println("Simple Interest = "+result);
    } catch (Exception ex) {}
    %>
    <%-- end web service invocation --%><hr/>
     <%-- start web service invocation --%><hr/>
    <%
    try {
            pack.NewWebService_Service service = new pack.NewWebService_Service();
            pack.NewWebService port = service.getNewWebServicePort();
        String p = request.getParameter("cp");
        String n = request.getParameter("cn");
```

```
        String r = request.getParameter("cr");
        String t = request.getParameter("ct");
            float pp = Float.parseFloat(p);
            float nn = Float.parseFloat(n);
        float rr = Float.parseFloat(r);
        float tt = Float.parseFloat(t);
            float result = port.cInterest(pp, nn, rr, tt);
            out.println("Compound Interest = "+result);
    } catch (Exception ex) {
    }
    %>
    <%-- end web service invocation --%><hr/>
    </body>
</html>
```

OUTPUT:

Server:

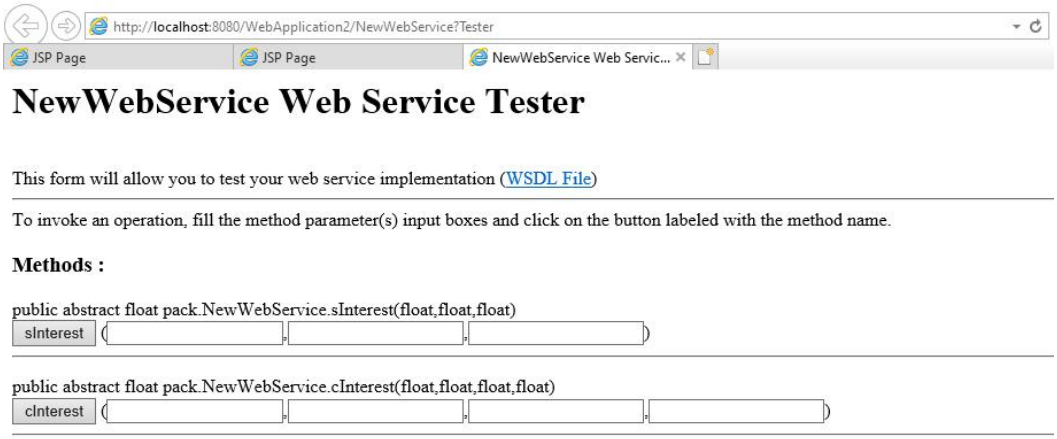


Fig 4.1 Web Service Index Page

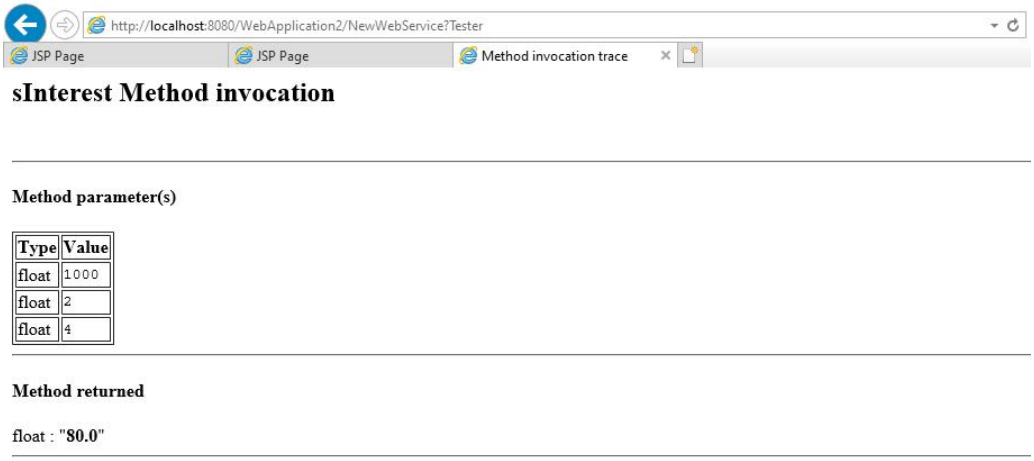


Fig 4.2 Simple Interest Invocation Page

Fig 4.3 Compound Interest Invocation Page

Client:

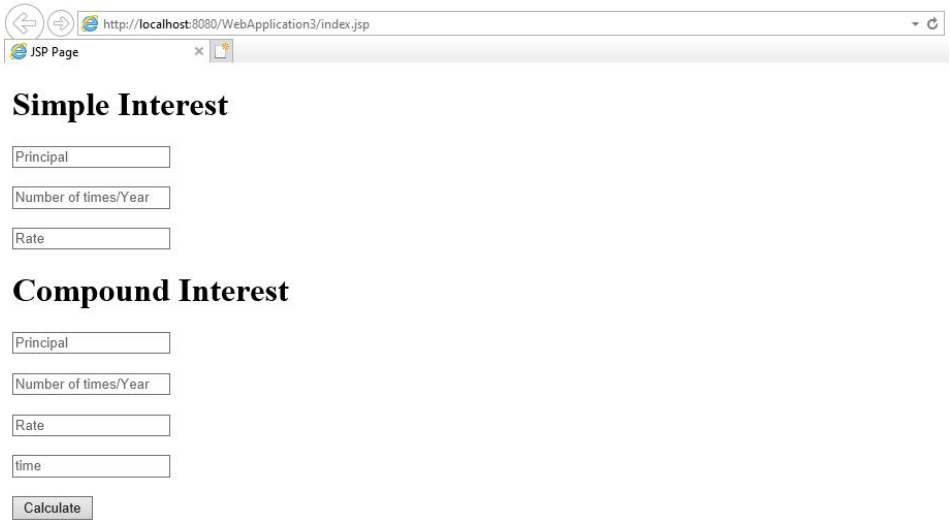


Fig 4.4 Client JSP Page

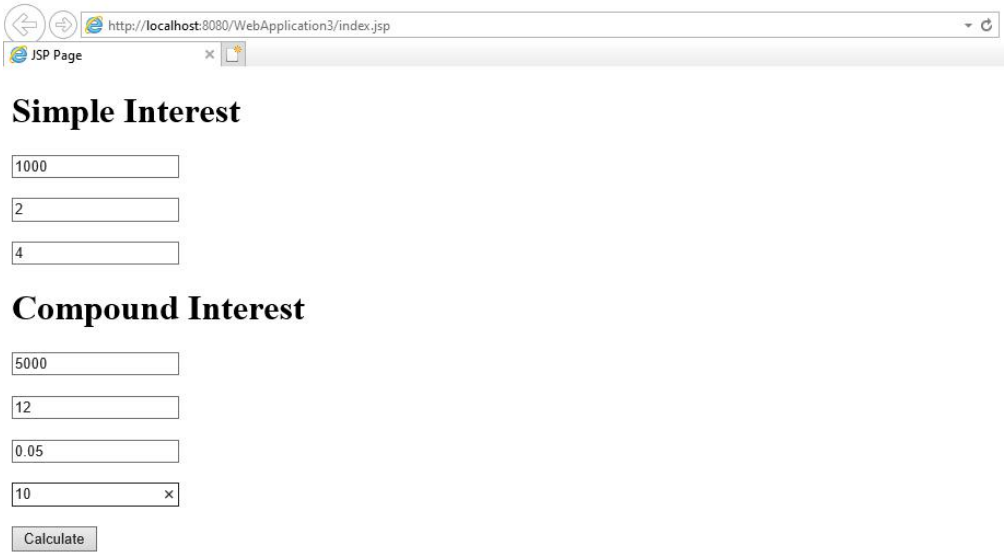


Fig 4.5 Client JSP Page with user data

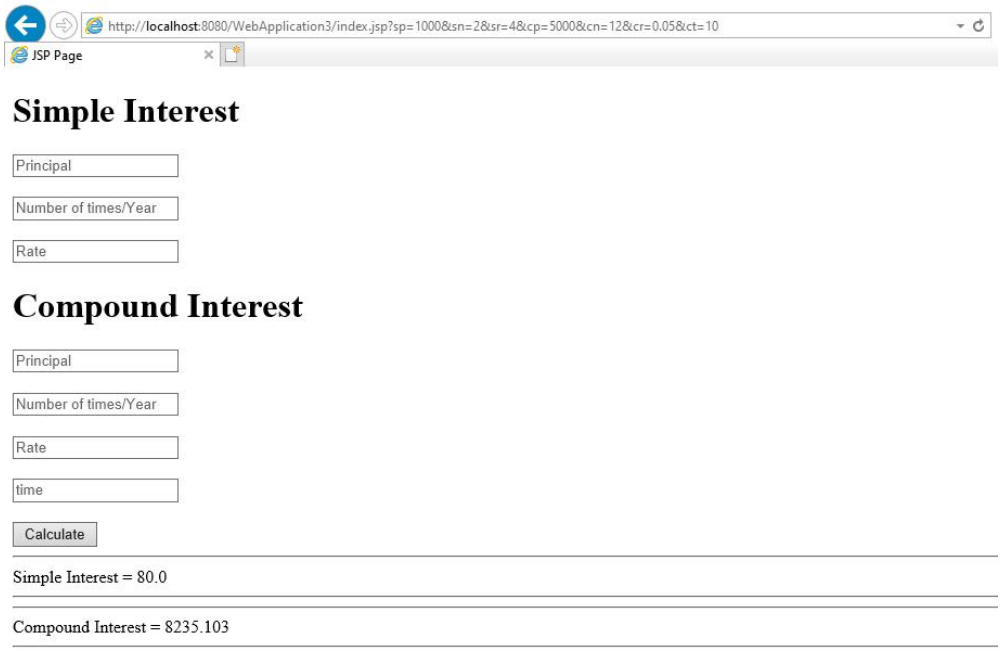


Fig 4.6 Client JSP Page with Output

RESULT:

Thus, Web Services was used to compute Simple and Compound Interest in JAVA.

Ex. No: 5
Date: 25/09/20

Web Services using Database Connectivity

AIM:

To implement client and server with database using SOAP Architecture

PROCEDURE:

Server:

1. Open NetBeans
2. Go to New > Project > Java Web > Web Application
3. Enter Project name and finish the creation process
4. Right click on the Project, go to New > Web Service
5. Provide Service and Package name and finish the creation process.
6. In the .java file, Go to Design View and click on "Add Operation"
7. Provide a name for the Operation and choose the number of parameters needed along with the data type, check mark the final and give OK
8. Go to Windows > Services > DataBase > JavaDB
9. Right Click and Choose Create Database, name it and complete the creation process
10. Right Click on the created database and choose connect
11. Under jdbc driver, Right click on the Table and add table. Provide the table name and the Columns Required and give Ok.
12. Right click on the created table name and select execute command, a SQL file opens, in that type the required sql commands and run the file
13. Under Projects>Web Services>.java file > Source Mode type the code required for database connectivity and presenting the data fetched in the web service
14. Right click on the Project and choose clean and build
15. Again right click on the Project and choose deploy
16. Right click on the Web Services name and choose Test Web Services

Client:

11. Open NetBeans
12. Go to New > Project > Java Web > Web Application
13. Enter Project name and finish the creation process
14. Right click on the Project, go to New > Web Service Client
15. Browse the Project name, and choose the server's web service name and choose Finish
16. In the .html page type the code to create a form and fetch data from user
17. Right click on the Source Packages and add Servlet file
18. Under Web Service Reference>Web Service> Port > function, drag and drop the function in the servlet
19. Type the code required to pass the data fetched from the user to the function under processRequest class
20. Run the HTML file

CODE:

Server:

```
package soalab;
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Vector;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author Moni
 */
@WebService(serviceName = "New")
public class New {

    /**
     * Web service operation
     */
    @WebMethod(operationName = "getid")
    public Vector getid(@WebParam(name = "id") final int id) {
        Vector v=new Vector();
        try
        {
            Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
            Connection con
=DriverManager.getConnection("jdbc:derby://localhost:1527/sample;create=true;user=app;password=mo
nica;");
            Statement st=con.createStatement();
            String sql="Select NAME,DEPARTMENT from COLLEGE where ID="+id+"";
            ResultSet rs = st.executeQuery(sql);
            if(rs!= null){
            while(rs.next())
            {
                v.addElement(rs.getString(1));
                v.addElement(rs.getString(2));
            }
            }
            else{
                v.addElement("Null");
            }
        }
        catch(Exception e)
        {
            System.err.println(e.getMessage());
            v.addElement(e.getMessage());
            return v;
        }
        return v;
    }
}
```

Database:

```
insert into APP.COLLEGE values (1,'Monica','Information Technology');
insert into APP.COLLEGE values (2,'Sonia','Material Science');
```

Client:

(HTML)

```
<!DOCTYPE html>
<html>
   <head>
      <title>TODO supply a title</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
   <body>
      <form action="NewServlet">
         <br>
            Enter Employee Id:    <input type="text" name="id" placeholder="id"><br><br>
                            <input type="submit" value="getid">
      </form>
   </body>
</html>
```

(Servlet)

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Iterator;
import java.util.Vector;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.ws.WebServiceRef;
import soalab.New_Service;

/**
 *
 * @author Moni
 */
public class NewServlet extends HttpServlet {
   @WebServiceRef(wsdlLocation = "WEB-INF/wsdl/localhost_8080/data/New.wsdl")
   private New_Service service;
   protected void processRequest(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      try (PrintWriter out = response.getWriter()) {
         int id =Integer.parseInt(request.getParameter("id"));
         Vector v=new Vector(getid(id));
         out.println("<!DOCTYPE html>");
         out.println("<html>");
         out.println("<head>");
         out.println("<title>Servlet NewServlet</title>");
```

```
        out.println("</head>");
        out.println("<h1>Database information</h1>");
        Iterator itr =v.listIterator();
        while(itr.hasNext())
        {
        out.println("<body>");
        out.println("<br>");
        out.println("<table                                                    border=1
cellspacing=1><tr><td>ID</td><td>Name</td><td>Department</td></tr>"                    +
"<tr><td>"+id+"</td><td>"+itr.next()+"</td><td>"+itr.next()+"</td></tr></table>");
        out.println("</body>");
        out.println("</html>");
        }

      }
   }
   @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      processRequest(request, response);
   }
   @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
      processRequest(request, response);
   }
   @Override
   public String getServletInfo() {
      return "Short description";
   }// </editor-fold>
   private java.util.List<java.lang.Object> getid(int id) {
      soalab.New port = service.getNewPort();
      return port.getid(id);
   }
}
```

OUTPUT:

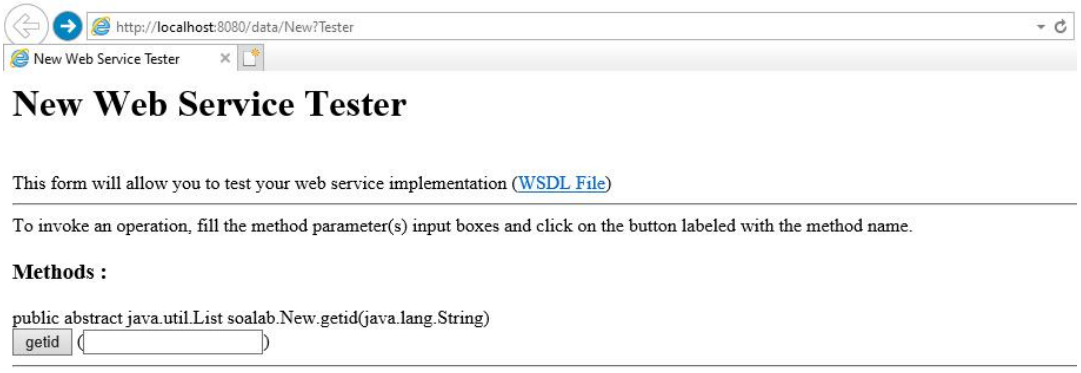


Fig 5.1 Web Service Page

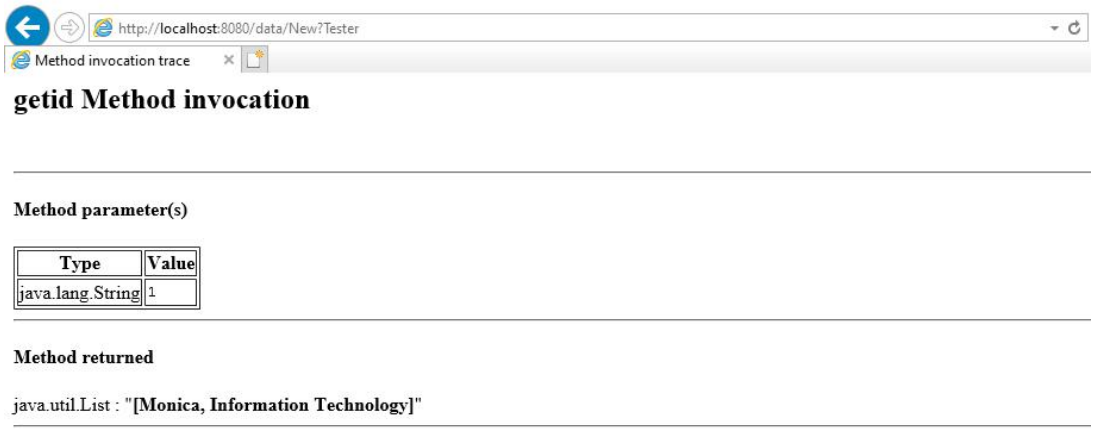


Fig 5.2 Web Service Invocation Page

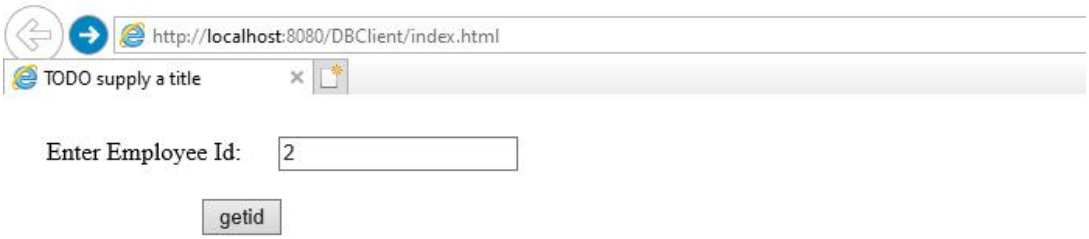


Fig 5.3 Client HTML Page

Fig 5.4 Client Page displaying data from Database

RESULT:

Thus, the client and server with Database was implemented using the SOAP Architecture.

Ex. No: 6
Date: 26/09/20

Encryption and Decryption using RSA Algorithm

AIM:

To implement XML Encryption and Decryption using RSA Algorithm

PROCEDURE:

1. Open Microsoft Visual Studio 2019
2. Go to New -> Project and select Windows Form Application
3. Enter name and finish the creation process.
4. In the Design View of the cs file, drag and drop the required text boxes, buttons and labels
5. In the Source View, type the code required to do the encryption and decryption
6. Run the Application

CODE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Security.Cryptography;

namespace WindowsFormsApp4
{
   public partial class Form1 : Form
   {
      UnicodeEncoding ByteConverter = new UnicodeEncoding();
      RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
      byte[] plaintext;
      byte[] encryptedtext;
      static public byte[] Encryption(byte[] Data, RSAParameters RSAKey, bool DoOAEPPadding)
      {
         try
         {
            byte[] encryptedData;
            using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())
            {
               RSA.ImportParameters(RSAKey);
               encryptedData = RSA.Encrypt(Data, DoOAEPPadding);
            }
            return encryptedData;
         }
         catch (CryptographicException e)
         {
            Console.WriteLine(e.Message);
            return null;
```

```
            }
        }

        static public byte[] Decryption(byte[] Data, RSAParameters RSAKey, bool DoOAEPPadding)
        {
            try

            {
                byte[] decryptedData;
                using (RSACryptoServiceProvider RSA = new RSACryptoServiceProvider())
                {
                    RSA.ImportParameters(RSAKey);
                    decryptedData = RSA.Decrypt(Data, DoOAEPPadding);
                }
                return decryptedData;
            }

            catch (CryptographicException e)
            {
                Console.WriteLine(e.ToString());
                return null;
            }
        }

        public Form1()

        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            plaintext = ByteConverter.GetBytes(textBox1.Text);
            encryptedtext = Encryption(plaintext, RSA.ExportParameters(false), false);
            textBox2.Text = ByteConverter.GetString(encryptedtext);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            byte[] decryptedtext = Decryption(encryptedtext, RSA.ExportParameters(true), false);
            textBox3.Text = ByteConverter.GetString(decryptedtext);
        }
    }
}
```

OUTPUT:

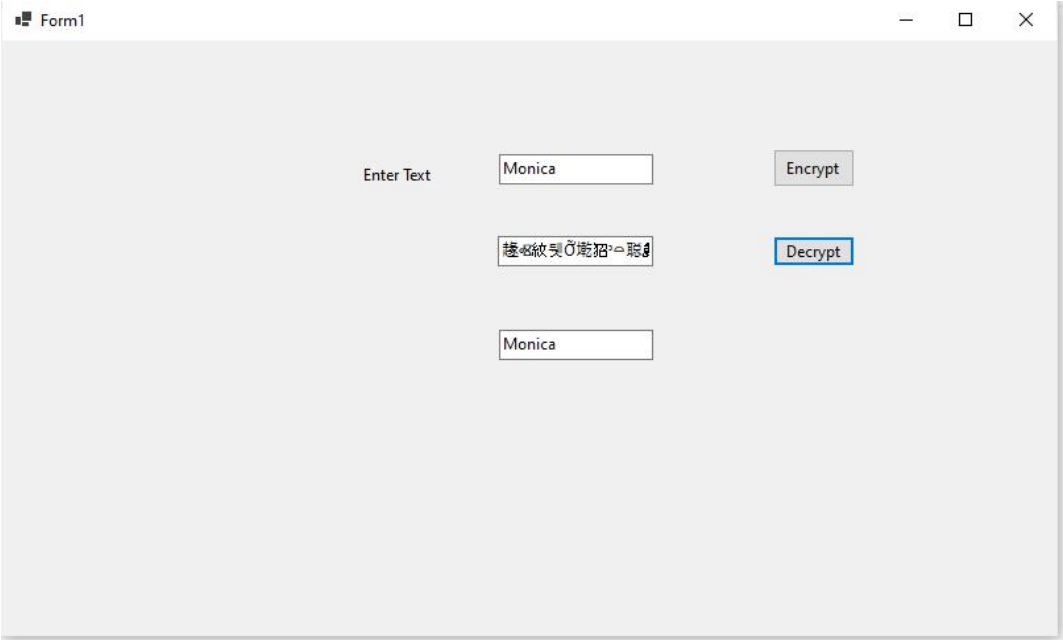


Fig 6.1 Encryption and Decryption Form

RESULT:

Thus, the XML Encryption and Decryption was implemented in Microsoft Visual Studio using RSA Algorithm

Ex. No: 7
Date: 03/10/20

XML-RPC Client and Server Implementation

AIM:

To implement XML-RPC Client and Server in Java

PROCEDURE:

Server:

1. Open NetBeans
2. Go to File -> New Project -> Java -> Java Application
3. Name the application and complete the creation process
4. Under the created Project folder, right click the libraries folder and select Add Jar/Folder and add the commons-codec.jar file and xmlrpc.jar file
5. In the .java file type the required code for creating a server and for providing the required function

Client:

1. Open NetBeans
2. Go to File -> New Project -> Java -> Java Application
3. Name the application and complete the creation process
4. Under the created Project folder, right click the libraries folder and select Add Jar/Folder and add the commons-codec.jar file and xmlrpc.jar file
5. In the .java file type the required code for creating a client and accessing the function from the server
6. Run the server and the the client

CODE:

Server:

```
package newempty;
/**
 *
 * @author Moni
 */
import org.apache.xmlrpc.*;
public class NewEmpty
{
   public Integer sum(int x,int y)
       {
          return x+y;
       }
 public static void main (String[] args)
 {
    try{
        System.out.println("Attempting to start XML-RPC Server...");
        WebServer server =new WebServer(80);
        server.addHandler("sample",new NewEmpty());
        server.start();
        System.out.println("Started successfully.");
        System.out.println("Accepting requests. (Halt program to stop.)");
```

```
    }catch(Exception exception){
        System.err.println("JavaServer: "+ exception);
    }
 }
}
```

Client:

```
package clientrpc;
/**
 *
 * @author Moni
 */
import java.io.IOException;
import java.util.*;
import org.apache.commons.codec.*;
import org.apache.xmlrpc.*;
public class ClientRPC {
   public static void main(String[] args) {
       try{
       XmlRpcClient server =new XmlRpcClient("http://localhost/RPC2");
          Vector params=new Vector();
          params.addElement(17);
          params.addElement(13);
          Object result = server.execute("sample.sum",params);
          int sum =((Integer) result);
          System.out.println("The sum is: "+ sum);
       }catch(IOException | XmlRpcException exception){
          System.err.println("JavaClient: "+ exception);
       }
   }
}
```

OUTPUT:

Server:

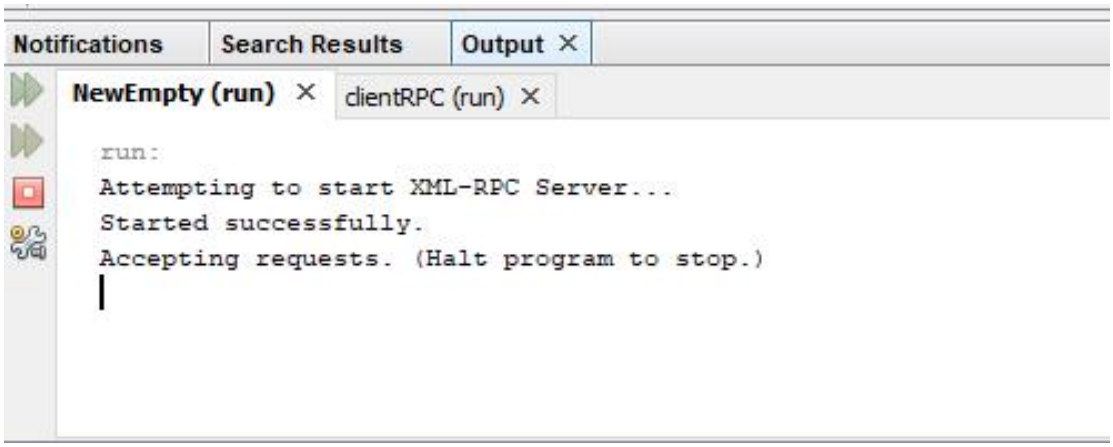


Fig 7.1 Server Output

Client:

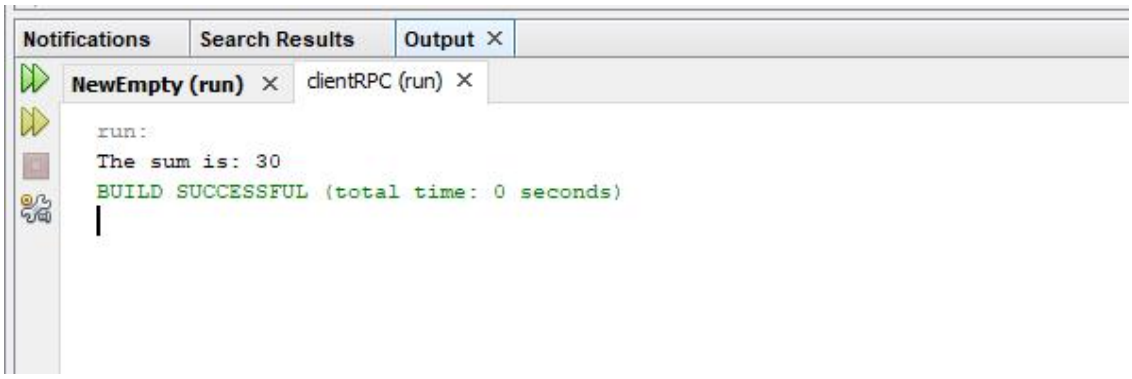


Fig 7.2 Client Output

RESULT:

Thus, the XML-RPC Client and Server was implemented in Java

Ex. No: 8
Date: 06/10/20

BPEL Implementation

AIM:

To implement BPEL application using NetBeans.

PROCEDURE:

BPEL Application:

1. Open NetBeans version below 6.5.
2. Go to New -> Project -> SOA and select BPEL Module and click next.
3. Enter name of the module finish the creation process.
4. Right click on the BPEL module and choose New and select WSDL Document.
5. Enter name of the document and click next.
6. Specify the input and output and click next.
7. Set the binding type as SOAP and click finish. A WSDL document will be created.
8. Right click on the BPEL module and choose New and select BPEL Process.
9. Enter name of the document and click finish.
10. Drag and drop wsdl file from project structure into the bpel file to create a partner link.
11. Enter the name of the partner link and click ok.
12. From the palette window drag and drop Receive, Assign and Reply services into the bpel process.
13. Connect Receive and Reply with the partner link.
14. Double click receive and reply and create respective input and output variables.
15. Double click assign and create any expression and connect input and output variables to the expression.
16. Right click the bpel file and select validate XML to validate the module.
17. BPEL module can be deployed using a composite application.

Composite Application:

1. Go to New -> Project -> SOA and select Composite Application and click next.
2. Enter name the finish the creation process
3. Right click the project and select the BPEL module and click add project.
4. Right click the project and build the composite application.
5. Right click the project and click deploy.
6. To test the application, right click the test folder inside the project and select new test case.
7. Enter the test case name and click next.
8. Select the WSDL file from the bpel module click next and select the operation to be tested and click finish.
9. Enter the input in the tag with input variable name.
10. Right click the test case and select run to get the output.

CODE:

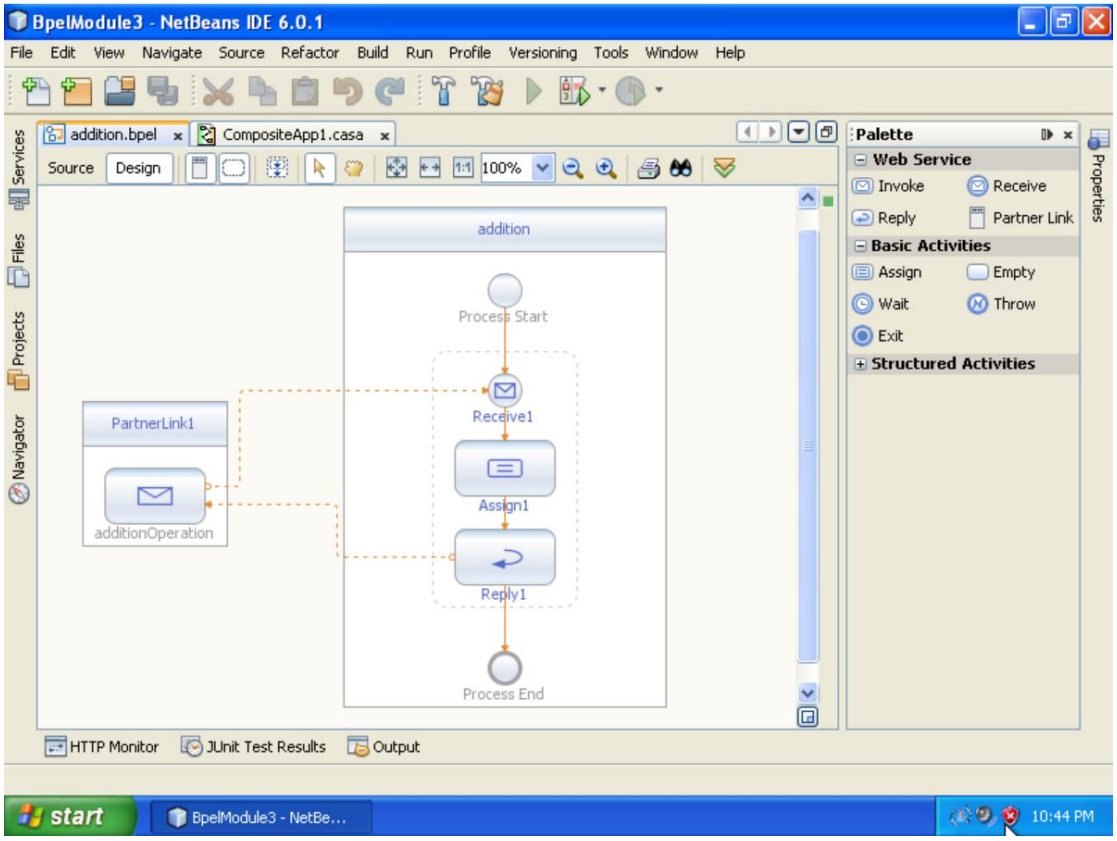


Fig 8.1 BPEL Module

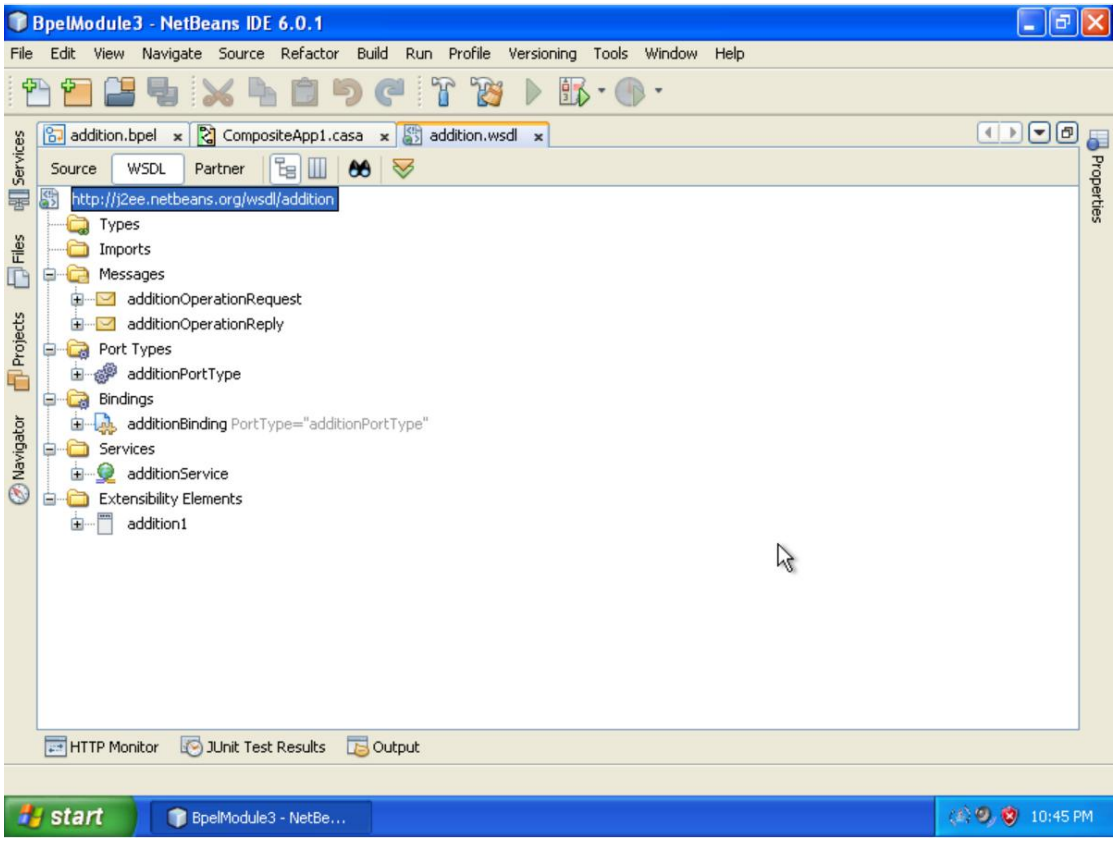


Fig 8.2 WSDL Structure

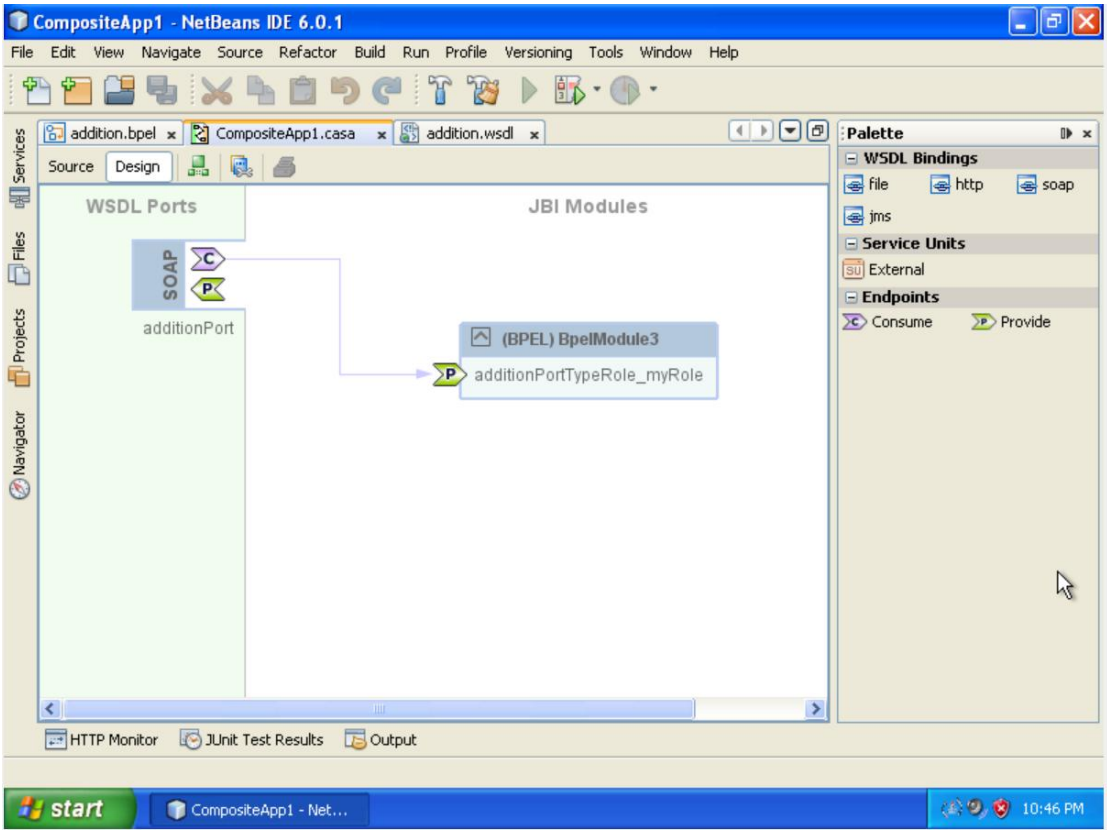


Fig 8.3 Composite Application

OUTPUT:

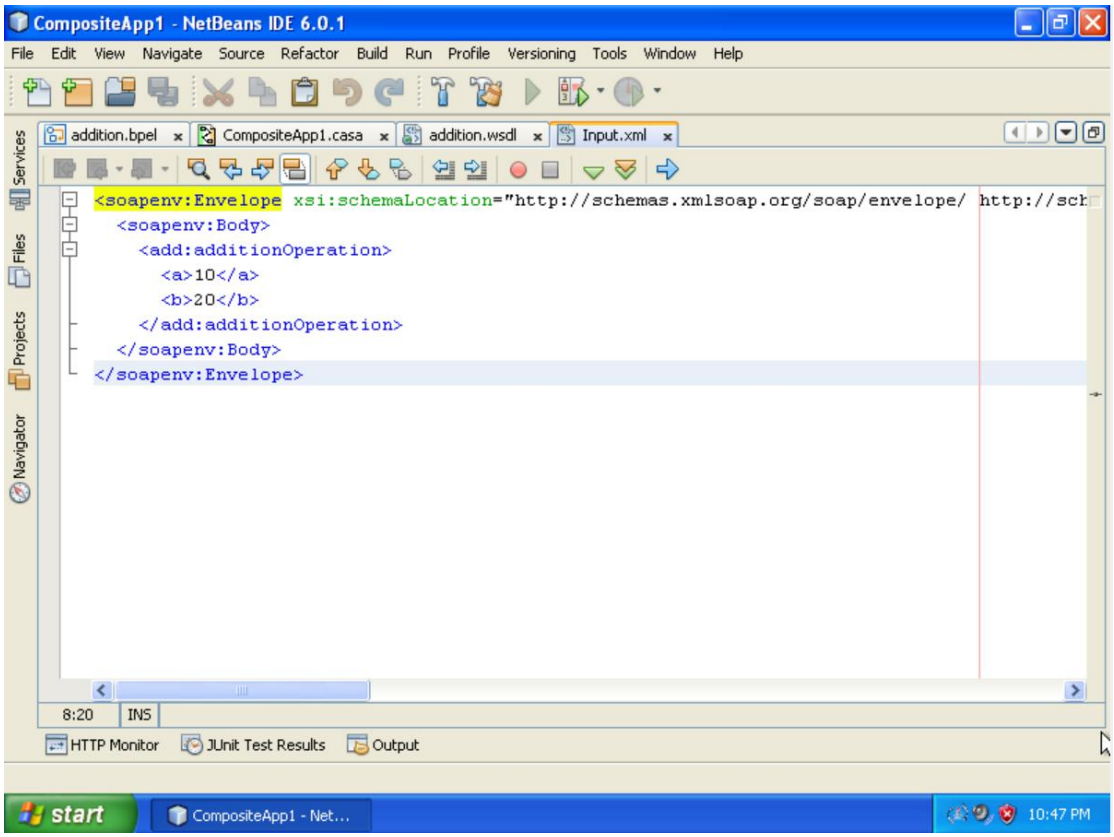


Fig 8.4 Addition input

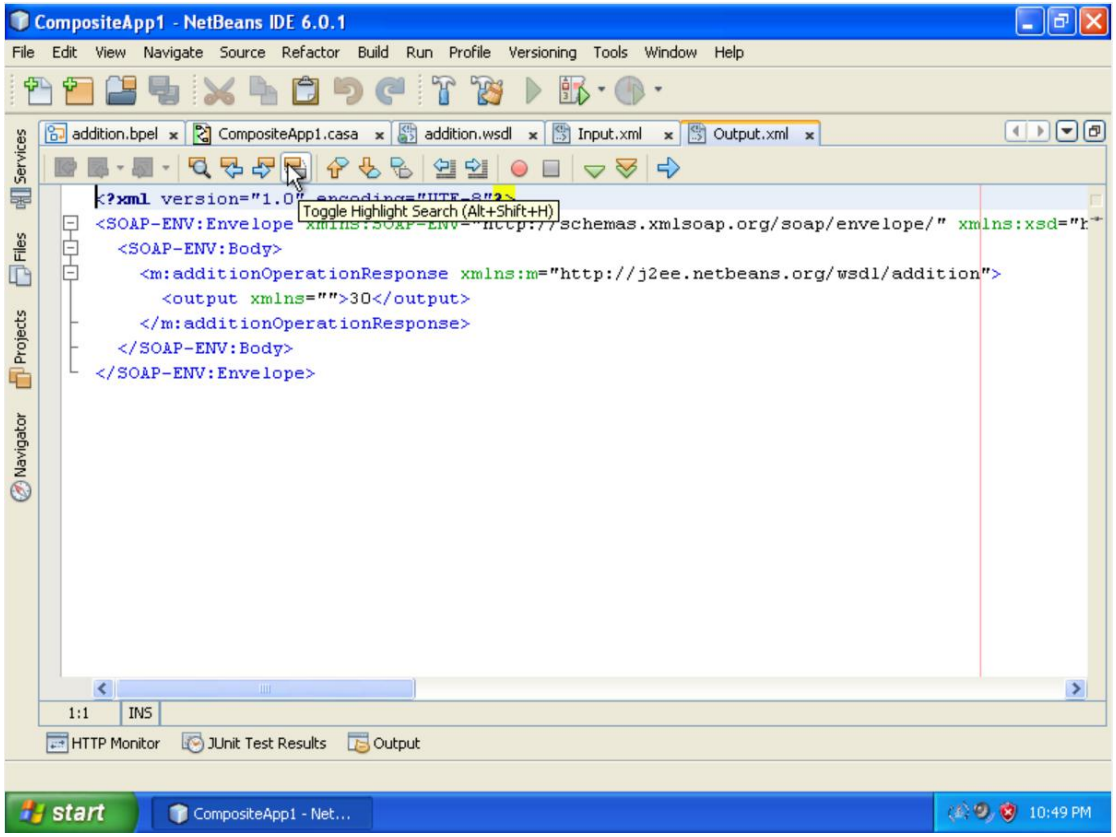


Fig 8.5 Addition Output

RESULT:

Thus, BPEL Module for addition of two numbers is done successfully and deployed in a composite application using NetBeans