NAAC Accredited "A" Grade

ISO 9001:2000

Best College
UNIVERSITY of MUMBAI

RAMNIRANJAN JHUNJHUNWALA COLLEGE

GHATKOPAR (W), MUMBAI - 400 086

DEPARTMENT OF INFORMATION TECHNOLOGY

2020 - 2021

M.Sc.(I.T.) SEM I

Distributed System

Name:Surekha Omprakash Rajbhar

Roll No.: 12

Hindi Vidya Prachar Samiti's

RAMNIRANJAN JHUNJHUNWALA COLLEGE (AUTONOMOUS)

Opposite Ghatkopar Railway Station, Ghatkopar West, Mumbai-400086

CERTIFICATE

This is to certify that Mr./Miss/Mrs. **Surekha Rajbhar** with Roll No.**02** has successfully completed the necessary course of experiments in the subject of **DISTRIBUTED SYSTEM** during the academic year **2020 – 2021** complying with the requirements of **RAMNIRANJAN JHUNJHUNWALA COLLEGE OF ARTS, SCIENCE AND COMMERCE**, for the course of **M.Sc. (IT)** semester -I.

Internal Examiner Date: **25-02-2021**

INDEX:

Prac no.	**Aim**	**Date**
1	Write a program for implementing Client Server communication model	26/11/2020
2	Write a program to show the object communication using RMI	05/02/2021

3	Show the implementation of Remote Procedure Call.	11/12/2020
4	Show the implementation of web services.	12/02/2021
5	Write a program to execute any one mutual exclusion algorithm	03/01/2021
6	Write a program to implement any one election algorithm.	10/01/2021
7	Show the implementation of any one clock synchronization algorithm	10/01/2021
8	Write a program to implement two phase commit protocol	18/02/2021

Practical 01

Aim: Write a program for implementing a Client Server communication model. .

Client Server communication model:

Client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A server host runs one or more server programs, which share their resources with clients. A client does not share any of its resources, but it requests content or service from a server. Clients therefore initiate communication sessions with servers, which await incoming requests. Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

The client-server characteristic describes the relationship of cooperating programs in an application. The server component provides a function or service to one or many clients, which initiate requests for such services. Servers are classified by the services they provide. For example, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitutes a service.

Example 1A: A client server based program using TCP to find if the number entered is prime.

Program:

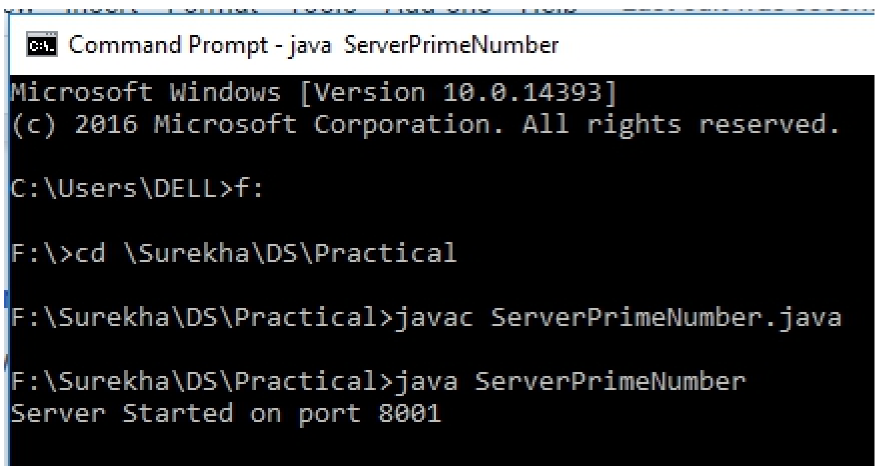
ServerPrimeNumber.java

```
import java.net.*;
import java.io.*;

public class ServerPrimeNumber {
    public static void main(String[] args) {
        try {
            final int PORT = 8001;
            ServerSocket ss = new ServerSocket(PORT);
            System.out.println("Server Started on port " + PORT);
            Socket s = ss.accept();
            DataInputStream dataInput = new DataInputStream(s.getInputStream());
            int x = dataInput.readInt();
            DataOutputStream dataOutput = new DataOutputStream(s.getOutputStream());
            int y = x / 2;

            if (x == 1) {
                dataOutput.writeUTF(x + " is neither Prime nor composite");
                System.exit(0);
            }

            boolean isPrime = false;
            for (int i = 2; i <= y; i++) {
                if (x % i == 0) {
                    isPrime = true;
                    break;
                }
            }

            if (isPrime) {
                dataOutput.writeUTF(x + " is not a Prime Number");
            } else {
                dataOutput.writeUTF(x + " is a Prime Number");
            }
            ss.close();
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

ClientPrimeNumber.java

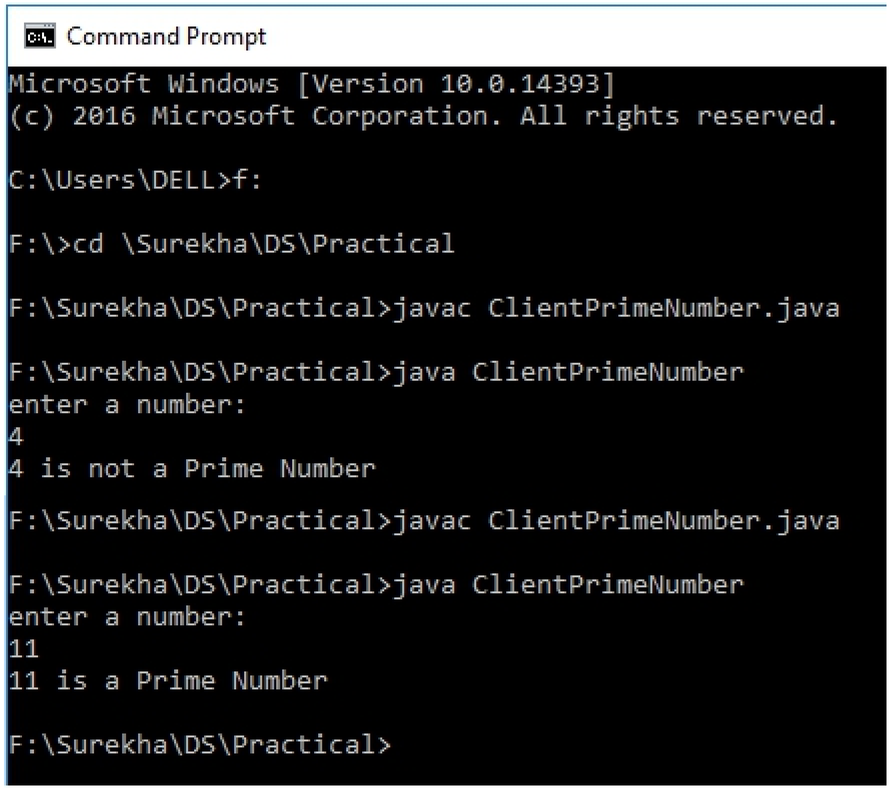
```
import java.net.*;
import java.io.*;

public class ClientPrimeNumber {
    public static void main(String[] args) {
        try {
            final int PORT = 8001;
            Socket s = new Socket("Localhost", PORT);
            BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter a number: ");
            int x = Integer.parseInt(input.readLine());
            DataOutputStream out = new DataOutputStream(s.getOutputStream());
            out.writeInt(x);
            DataInputStream in = new DataInputStream(s.getInputStream());
            System.out.println(in.readUTF());
            s.close();
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

output :



Command Prompt - java ServerPrimeNumber

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac ServerPrimeNumber.java

F:\Surekha\DS\Practical>java ServerPrimeNumber
Server Started on port 8001
```



Command Prompt

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac ClientPrimeNumber.java

F:\Surekha\DS\Practical>java ClientPrimeNumber
enter a number:
4
4 is not a Prime Number

F:\Surekha\DS\Practical>javac ClientPrimeNumber.java

F:\Surekha\DS\Practical>java ClientPrimeNumber
enter a number:
11
11 is a Prime Number

F:\Surekha\DS\Practical>
```

Example 1B: A client server TCP based chatting application.

Program:

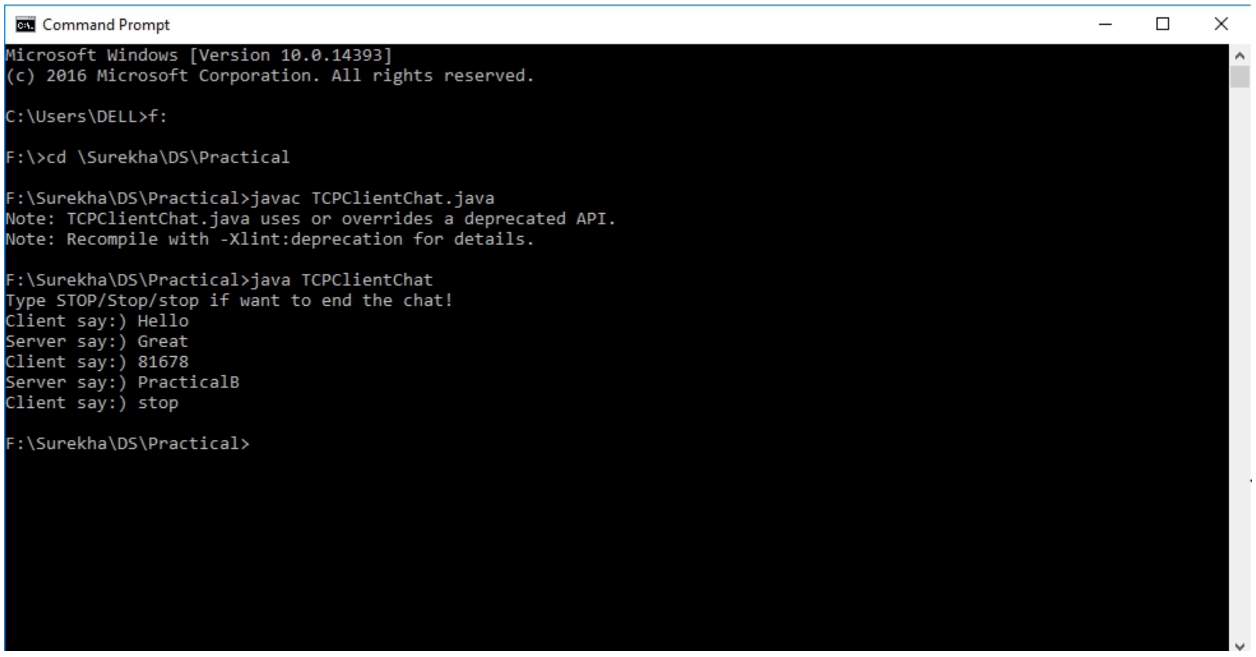
TCPServerChat.java

```
public class TCPServerChat {
    public static void main(String args[]) {
        try {

            final int PORT = 8001;
            ServerSocket serverSocket = new ServerSocket(PORT);

            System.out.println("Server Ready For Chat on PORT " + PORT);

            Socket socket = serverSocket.accept();
            BufferedReader inputBReader = new BufferedReader(new
InputStreamReader(System.in));

            DataOutputStream outputStream = new DataOutputStream(socket.getOutputStream());
            DataInputStream inputStream = new DataInputStream(socket.getInputStream());
            String receive;
            String send;

            while ((receive = inputStream.readLine()) != null) {
                if (receive.toLowerCase().equals("stop"))
                    break;
                System.out.println("Client Says: " + receive);
                System.out.print("Server say:) ");
                send = inputBReader.readLine();
                outputStream.writeBytes(send + "\n");
            }

            inputBReader.close();
            inputStream.close();
            outputStream.close();
            serverSocket.close();

        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

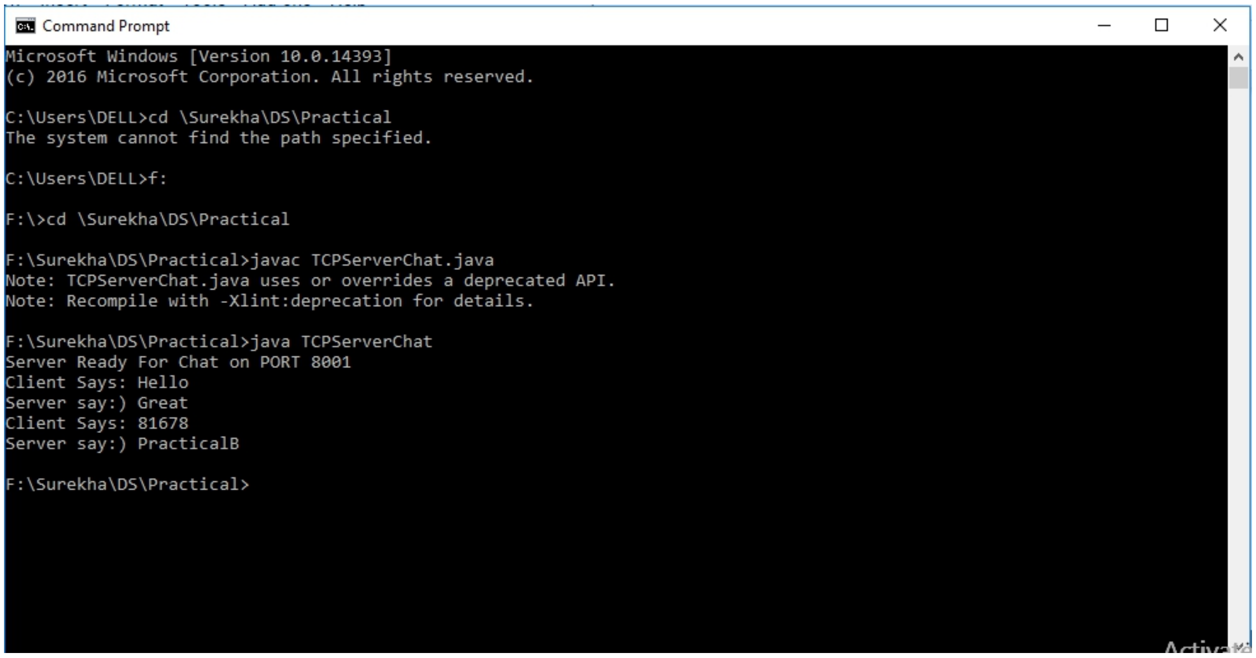
TCPClientChat.java

```
import java.net.*;
import java.io.*;

public class TCPClientChat {
    public static void main(String args[]) {
        try {
            final int PORT = 8001;
            final String HOST = "LocalHost";
            Socket clientSocket = new Socket(HOST, PORT);
            BufferedReader inpBufferedReader = new BufferedReader(new
InputStreamReader(System.in));

            DataOutputStream outputStream = new
DataOutputStream(clientSocket.getOutputStream());
            DataInputStream inputStream = new DataInputStream(clientSocket.getInputStream());
            String send;
            System.out.println("Type STOP/Stop/stop if want to end the chat!");
            System.out.print("Client say:) ");
            while ((send = inpBufferedReader.readLine()) != null) {
                outputStream.writeBytes(send + "\n");
                if (send.toLowerCase().equals("stop"))
                    break;
                System.out.println("Server say:) " + inputStream.readLine());
                System.out.print("Client say:) ");
            }
            inpBufferedReader.close();
            inputStream.close();
            outputStream.close();
            clientSocket.close();
        } catch (Exception e) {
            System.out.println(e.toString());
        }
    }
}
```

Output:



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac TCPClientChat.java
Note: TCPClientChat.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

F:\Surekha\DS\Practical>java TCPClientChat
Type STOP/Stop/stop if want to end the chat!
Client say:) Hello
Server say:) Great
Client say:) 81678
Server say:) PracticalB
Client say:) stop

F:\Surekha\DS\Practical>
```



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>cd \Surekha\DS\Practical
The system cannot find the path specified.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac TCPServerChat.java
Note: TCPServerChat.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

F:\Surekha\DS\Practical>java TCPServerChat
Server Ready For Chat on PORT 8001
Client Says: Hello
Server say:) Great
Client Says: 81678
Server say:) PracticalB

F:\Surekha\DS\Practical>
```

Practical 02

Aim: Write a program to show the object communication using RM**I.**

RMI (Remote Method Invocation)

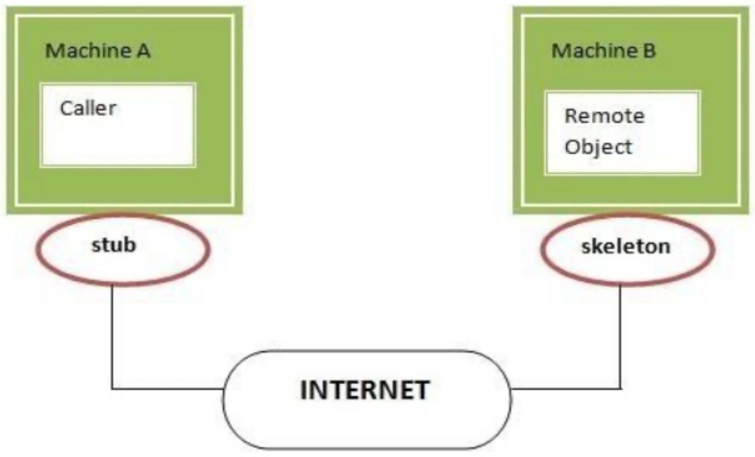
The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed applications in java. The RMI allows an object to invoke methods on an object running in another JVM. The RMI provides remote communication between the applications using two objects *stub* and *skeleton*. A **remote object** is an object whose method can be invoked from another JVM.

Stub : The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

Skeleton : The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.



2A: A RMI based application program to display current date and time.

Program:

ServerDate.java

```
import java.rmi.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.*;
import java.util.*;

public class ServerDate extends UnicastRemoteObject implements InterDate{

   public ServerDate() throws RemoteException
         {
          }
   public String display() throws RemoteException
   {
      String str = "";
      Date d = new Date();
      str = d.toString();
      return str;
   }

   public static void main(String args[]) throws RemoteException
   {
      try
      {
        ServerDate s1= new ServerDate();
        Registry reg = LocateRegistry.createRegistry(5678);
        reg.rebind("DS",s1);
        System.out.println("Object registed....");
      }
      catch(RemoteException e)
      {
        System.out.println("exception"+e);
      }
   }
}
```

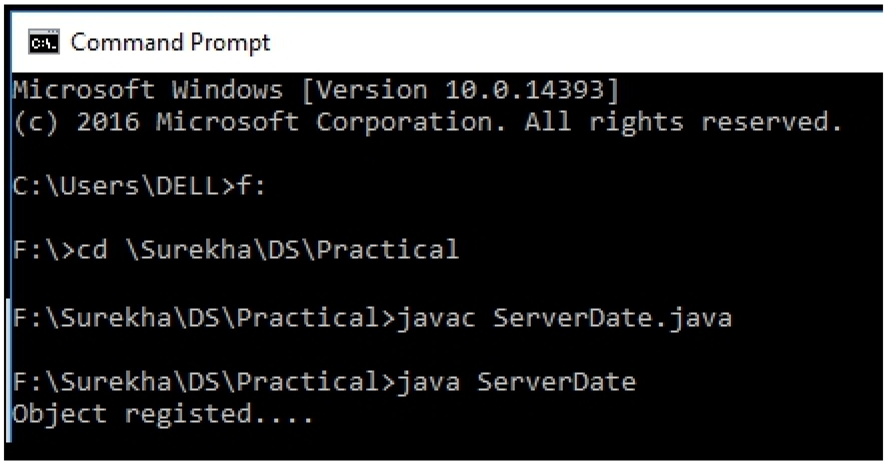
ClientDate.java

```
import java.rmi.*;
import java.io.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class ClientDate
{
    public static void main(String args[]) throws RemoteException
    {
        ClientDate c = new ClientDate();
        c.connectRemote();
    }
    private void connectRemote() throws RemoteException
    {
        try
            {
                String s1;
                Registry reg = LocateRegistry.getRegistry("localhost",5678);
                InterDate h1 = (InterDate)reg.lookup("DS");
                s1=h1.display();
                System.out.println(s1);
            }
                    catch(NotBoundException|RemoteException e)
        {
            System.out.println("exception"+e);
        }

    }
}
```

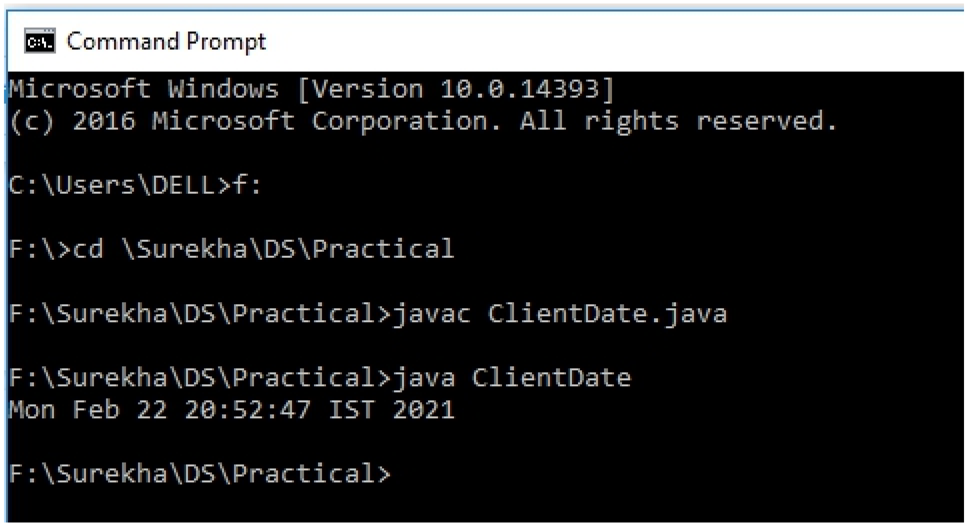
InterDate.java

```
import java.rmi.*;
public interface InterDate extends Remote
{
public String display() throws RemoteException;
}
```

Output:



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac ServerDate.java

F:\Surekha\DS\Practical>java ServerDate
Object registed....
```



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac ClientDate.java

F:\Surekha\DS\Practical>java ClientDate
Mon Feb 22 20:52:47 IST 2021

F:\Surekha\DS\Practical>
```

2B: A RMI based application program that converts digits to words, e.g. 123 will be converted to one two three

Program:

InterfaceConverter.java

```
import java.rmi.*;

public interface InterfaceConverter extends Remote {
    public String convertDigit(String no) throws RemoteException;;
}
```

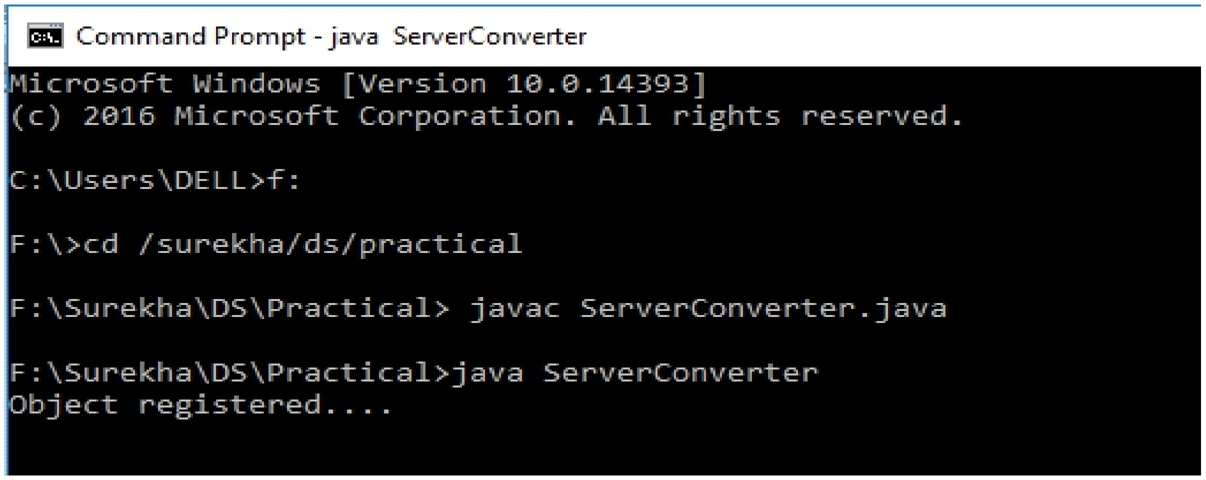
ClientConverter.java

```
import java.rmi.*;
import java.io.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

public class ClientConverter {
    public static void main(String args[]) throws RemoteException, IOException {
        ClientConverter c = new ClientConverter();
        c.connectRemote();
    }

    private void connectRemote() throws RemoteException, IOException {
        try {
            Registry reg = LocateRegistry.getRegistry("localhost", 5678);
            InterfaceConverter h1 = (InterfaceConverter) reg.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);
        } catch (NotBoundException | RemoteException e) {
            System.out.println("exception" + e);
        }
    }
}
```

ServerConverter.java

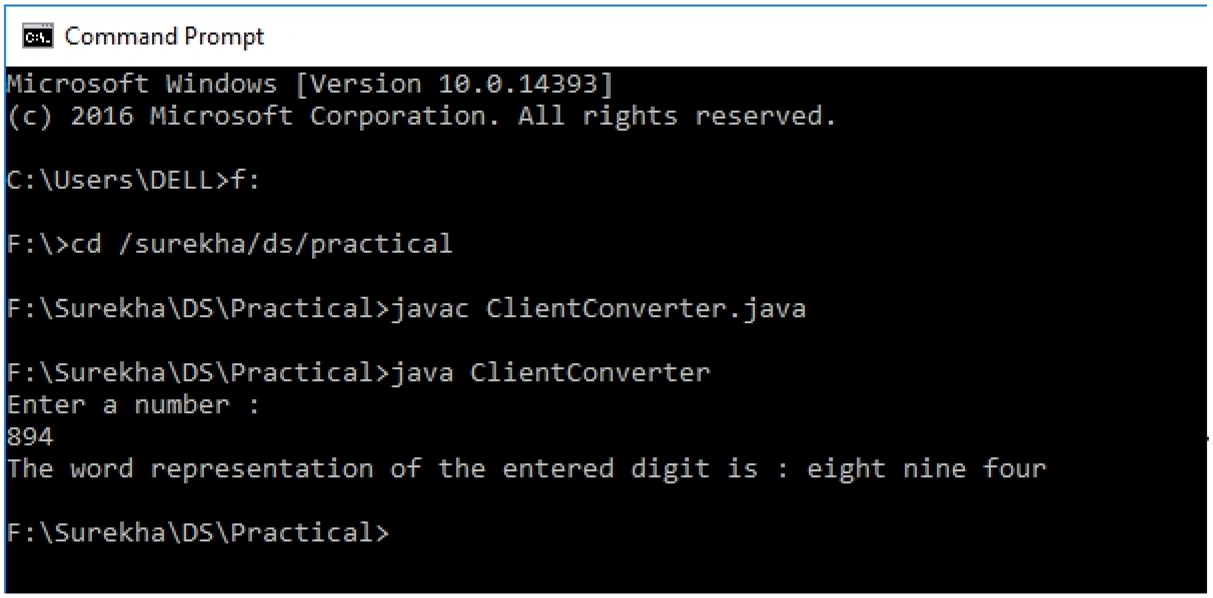
```
import java.rmi.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.*;

public class ServerConverter extends UnicastRemoteObject implements InterfaceConverter {
    public ServerConverter() throws RemoteException {
    }
    public String convertDigit(String no) throws RemoteException {
        String str = "";
        for (int i = 0; i < no.length(); i++) {
            int p = no.charAt(i);
            switch (p) {
                case 48:
                    str += "zero ";
                    break;
                case 49:
                    str += "one ";
                    break;
                case 50:
                    str += "two ";
                    break;
                case 51:
                    str += "three ";
                    break;
                case 52:
                    str += "four ";
                    break;
                case 53:
                    str += "five ";
                    break;
                case 54:
                    str += "six ";
                    break;
                case 55:
                    str += "seven ";
                    break;
                case 56:
                    str += "eight ";
                    break;
                case 57:
                    str += "nine ";
                    break;

            }
        }
        return str;
    }
    public static void main(String args[]) throws RemoteException {
        try {
            ServerConverter s1 = new ServerConverter();
            Registry reg = LocateRegistry.createRegistry(5678);
            reg.rebind("Wrd", s1);
            System.out.println("Object registered....");
        } catch (RemoteException e) {
            System.out.println("exception" + e);
        }
    }
}
```

Output:



Command Prompt - java ServerConverter

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd /surekha/ds/practical

F:\Surekha\DS\Practical> javac ServerConverter.java

F:\Surekha\DS\Practical>java ServerConverter
Object registered....
```



Command Prompt

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd /surekha/ds/practical

F:\Surekha\DS\Practical>javac ClientConverter.java

F:\Surekha\DS\Practical>java ClientConverter
Enter a number :
894
The word representation of the entered digit is : eight nine four

F:\Surekha\DS\Practical>
```

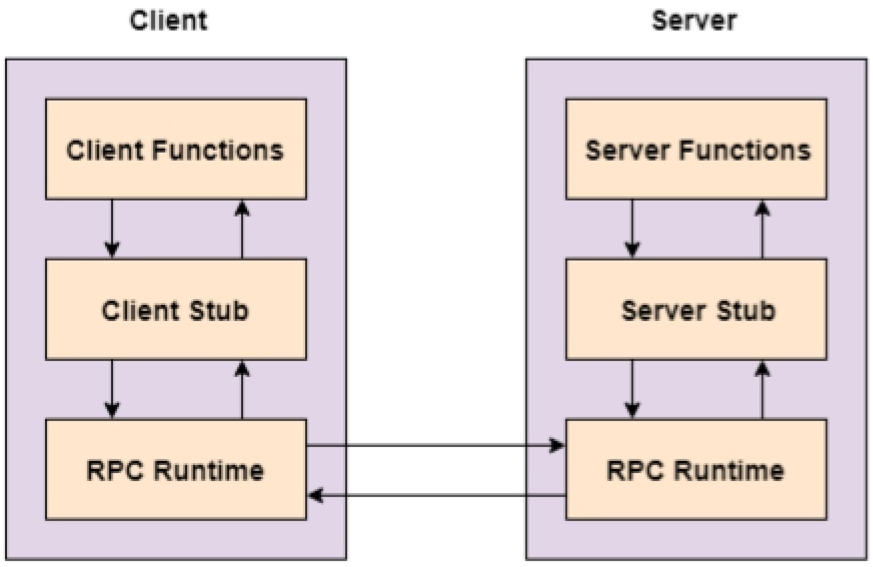
Practical 03

Aim: Show the implementation of Remote Procedure Call.

Remote Procedure Call (RPC):

A remote procedure call is an inter process communication technique that is used for clientserver-based applications. It is also known as a subroutine call or a function call.
A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request,

it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.
The sequence of events in a remote procedure call are given as follows:
• The client stub is called by the client
• The client stub makes a system call to send the message to the server and puts the parameters in the message.
• The message is sent from the client to the server by the client’s operating system
• The message is passed to the server stub by the server operating system.
• The parameters are removed from the message by the server stub. • Then, the server procedure is called by the server stub.
A diagram that demonstrates this is as follows:



The following steps take place during a RPC:
1. A client invokes a client stub procedure, passing parameters in the usual way. The client stub resides within the client’s own address space.
2. The client stub marshalls(pack) the parameters into a message. Marshalling includes converting the representation of the parameters into a standard format, and copying each parameter into the message.
3. The client stub passes the message to the transport layer, which sends it to the remote server machine.
4. On the server, the transport layer passes the message to a server stub, which demarshalls(unpack) the parameters and calls the desired server routine using the regular procedure call mechanism.

Example 3A: A program to implement simple calculator operations like addition, subtraction, multiplication and division.

Program :

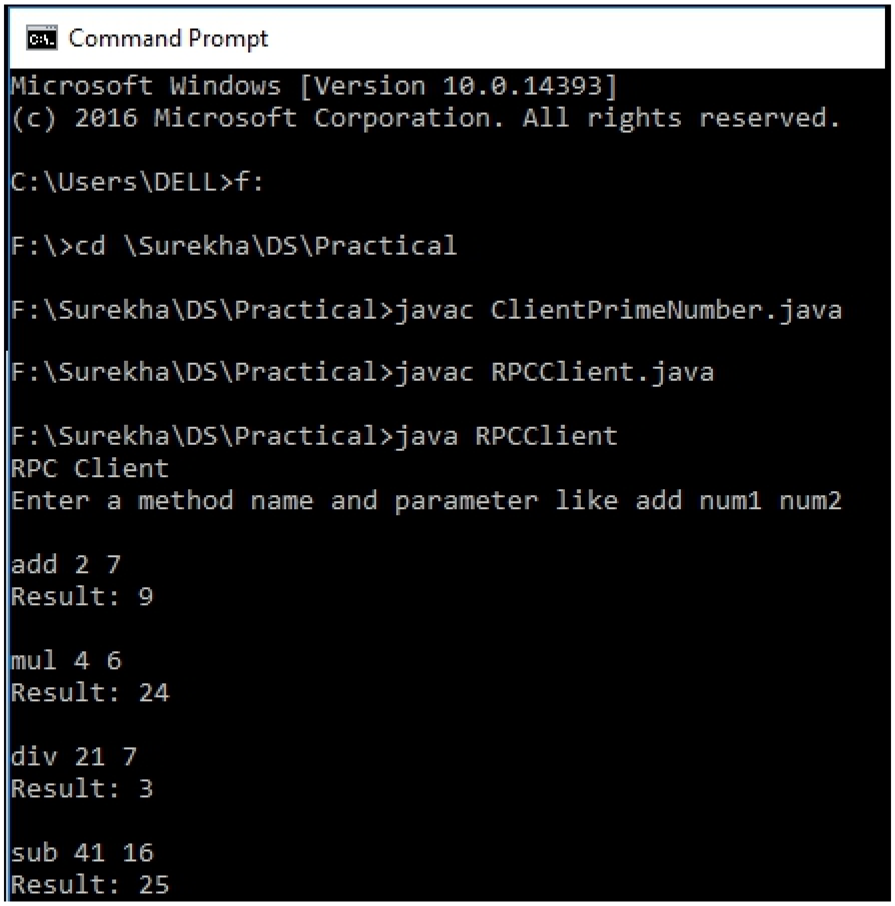
RCPServer.java

```
import java.io.*;
import java.net.*;
import java.util.StringTokenizer;

final class RPCServer {
   DatagramSocket ds;
   DatagramPacket dp;

   String str, methodName;

   int val1, val2, result;

   RPCServer() {
      try {
        final int PORT = 1200;

        ds = new DatagramSocket(PORT);

        byte b[] = new byte[4096];
        while (true) {
            dp = new DatagramPacket(b, b.length);
            ds.receive(dp);

            str = new String(dp.getData(), 0, dp.getLength());

            if (str.equalsIgnoreCase("q")) {
                System.exit(1);
            }

            StringTokenizer tokenizer = new StringTokenizer(str, " ");

            while (tokenizer.hasMoreTokens()) {
                String token = tokenizer.nextToken();
                methodName = token.toLowerCase();
                val1 = Integer.parseInt(tokenizer.nextToken());
                val2 = Integer.parseInt(tokenizer.nextToken());
            }
            System.out.println(str);

            InetAddress address = InetAddress.getLocalHost();
            final int DEST_PORT = 1300;

            switch (methodName) {
                case "add":
                    result = val1 + val2;
                    break;
                case "sub":
                    result = val1 - val2;
                    break;
                case "mul":
                    result = val1 * val2;
                    break;
                case "div":
                    result = val1 / val2;
                    break;
            }

            byte resultBytes[] = Integer.toString(result).getBytes();
            DatagramSocket datagramSocket = new DatagramSocket();

            DatagramPacket datagramPacket = new DatagramPacket(resultBytes,
resultBytes.length, address, DEST_PORT);

            System.out.println("Result: " + result + "\n");
            datagramSocket.send(datagramPacket);
            }
        }

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

    }

    public static void main(String[] args) {
        new RPCServer();
    }
}
```

RPCClient.java

```
import java.io.*;
import java.net.*;

public class RPCClient {
    RPCClient() {
        try {

            InetAddress address = InetAddress.getLocalHost();
            DatagramSocket datagramSocket = new DatagramSocket();
            DatagramSocket datagramSocketForServer = new DatagramSocket(1300);

            System.out.println("RPC Client");
            System.out.println("Enter a method name and parameter like add num1 num2\n");

            final int PORT = 1200;

            while (true) {
                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                String str = br.readLine();
                byte b[] = str.getBytes();
                DatagramPacket dp = new DatagramPacket(b, b.length, address, PORT);
                datagramSocket.send(dp);
                dp = new DatagramPacket(b, b.length);
                datagramSocketForServer.receive(dp);
                String s = new String(dp.getData(), 0, dp.getLength());
                System.out.println("Result: " + s + "\n");

            }

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

    public static void main(String[] args) {
        new RPCClient();
    }
}
```

Output -



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac ClientPrimeNumber.java

F:\Surekha\DS\Practical>javac RPCClient.java

F:\Surekha\DS\Practical>java RPCClient
RPC Client
Enter a method name and parameter like add num1 num2

add 2 7
Result: 9

mul 4 6
Result: 24

div 21 7
Result: 3

sub 41 16
Result: 25
```

Example 3B: A program that finds the square, square root, cube and cube root of the entered number.

Program:

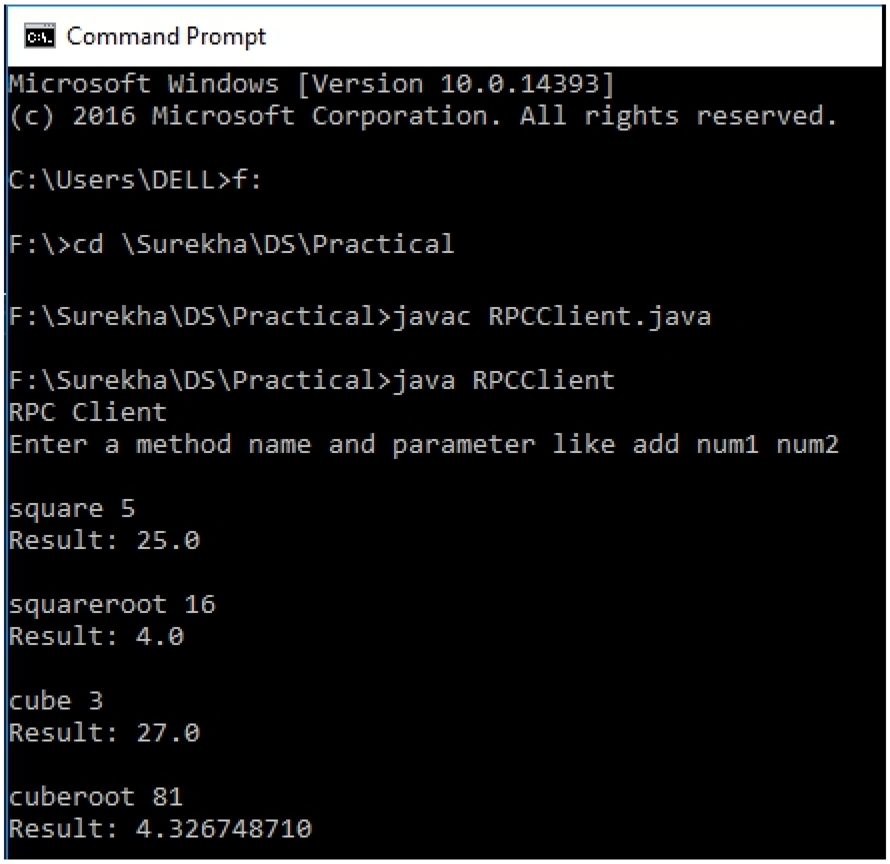
RCPServerB.java

```
import java.io.*;
import java.net.*;
import java.util.StringTokenizer;

final class RPCServerB {
   DatagramSocket ds;
   DatagramPacket dp;

   String str, methodName;

   double num, result;

   RPCServerB() {
      try {

         final int PORT = 1200;

         ds = new DatagramSocket(PORT);

         byte b[] = new byte[4096];
         while (true) {
            dp = new DatagramPacket(b, b.length);
            ds.receive(dp);

            str = new String(dp.getData(), 0, dp.getLength());

            if (str.equalsIgnoreCase("q")) {
               System.exit(1);
            }

            StringTokenizer tokenizer = new StringTokenizer(str, " ");

            while (tokenizer.hasMoreTokens()) {
               String token = tokenizer.nextToken();
               methodName = token.toLowerCase();
               num = Integer.parseInt(tokenizer.nextToken());
            }
            System.out.println(str);

            InetAddress address = InetAddress.getLocalHost();
            final int DEST_PORT = 1300;

            switch (methodName) {
                case "square":
                    result = num * num;
                    break;
                case "squareroot":
                    result = Math.sqrt(num);
                    break;
                case "cube":
                    result = num * num * num;
                    break;
                case "cuberoot":
                    result = Math.cbrt(num);
                    break;
            }

            byte resultBytes[] = Double.toString(result).getBytes();
            DatagramSocket datagramSocket = new DatagramSocket();

            DatagramPacket datagramPacket = new DatagramPacket(resultBytes,
resultBytes.length, address, DEST_PORT);

            System.out.println("Result: " + result + "\n");
            datagramSocket.send(datagramPacket);
        }
    }

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

}

public static void main(String[] args) {
    new RPCServerB();
}
}
```

RPCClient.java

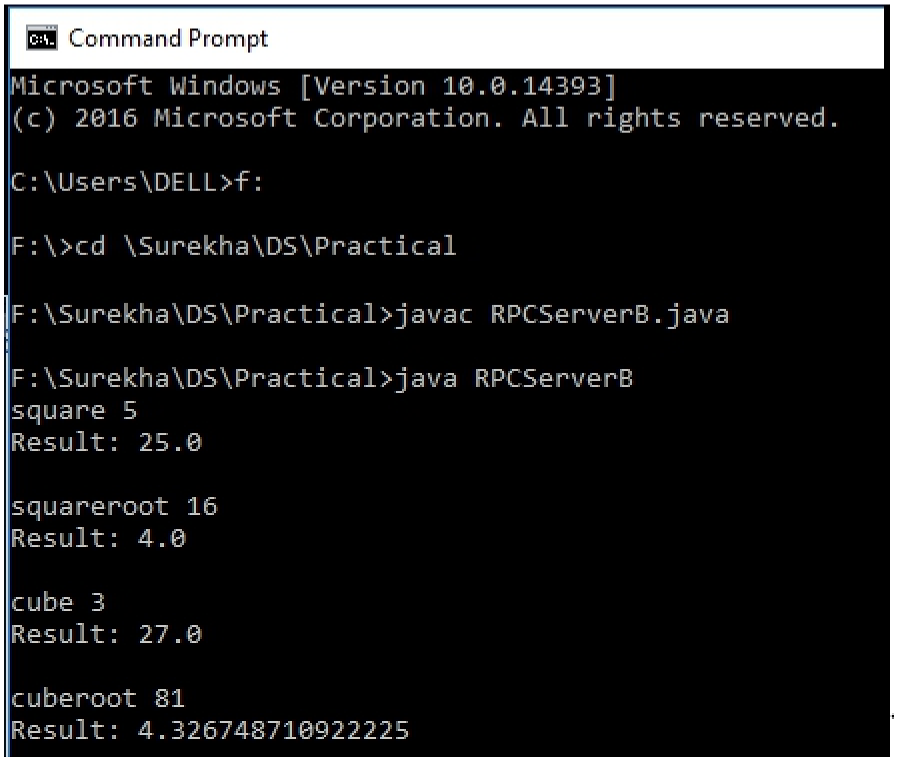
```
import java.io.*;
import java.net.*;

public class RPCClient {
   RPCClient() {
      try {

         InetAddress address = InetAddress.getLocalHost();
         DatagramSocket datagramSocket = new DatagramSocket();
         DatagramSocket datagramSocketForServer = new DatagramSocket(1300);

         System.out.println("RPC Client");
         System.out.println("Enter a method name and parameter like square num\n");

         final int PORT = 1200;

         while (true) {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            String str = br.readLine();
            byte b[] = str.getBytes();
            DatagramPacket dp = new DatagramPacket(b, b.length, address, PORT);
            datagramSocket.send(dp);
            dp = new DatagramPacket(b, b.length);
            datagramSocketForServer.receive(dp);
            String s = new String(dp.getData(), 0, dp.getLength());
            System.out.println("Result: " + s + "\n");

         }

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

   public static void main(String[] args) {
      new RPCClient();
   }
}
```

Output -



Command Prompt

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac RPCClient.java

F:\Surekha\DS\Practical>java RPCClient
RPC Client
Enter a method name and parameter like add num1 num2

square 5
Result: 25.0

squareroot 16
Result: 4.0

cube 3
Result: 27.0

cuberoot 81
Result: 4.326748710
```



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac RPCServerB.java

F:\Surekha\DS\Practical>java RPCServerB
square 5
Result: 25.0

squareroot 16
Result: 4.0

cube 3
Result: 27.0

cuberoot 81
Result: 4.326748710922225
```

Practical 04

Aim: Show the implementation of web services.
Web services:
A Web Service is a software program that uses XML to exchange information with other software via common internet protocols. In a simple sense, Web Services are a way of interacting with objects over the Internet.
A web service is
• Language Independent.
• Protocol Independent.
• Platform Independent.
• It assumes a stateless service architecture.
• Scalable (e.g. multiplying two numbers together to an entire customer-relationship management system).
• Programmable (encapsulates a task).
• Based on XML (open, text-based standard).
• Self-describing (metadata for access and use).
• Discoverable (search and locate in registries)- ability of applications and developers to search for and locate desired Web services through registries. This is based on UDDI.

Key Web Service Technologies
1. XML- Describes only data. So, any application that understands XML-regardless of the application's programming language or platform has the ability to format XML in a variety of ways (well-formed or valid). 2. SOAP- Provides a communication mechanism between services and applications. 3. WSDL- Offers a uniform method of describing web services to other programs. 4. UDDI- Enables the creation of searchable Web services registries.

Web Services Limitations
1. SOAP, WSDL, UDDI- require further development. 2. Interoperability. 3. Royalty fees. 4. Too slow for use in high-performance situations. 5. Increase traffic on networks. 6. The lack of security standards for Web services. 7. The standard procedure for describing the quality (i.e. levels of performance, reliability, security etc.) of particular Web services – management of Web services. 8. The standards that drive Web services are still in draft form (always will be in refinement).

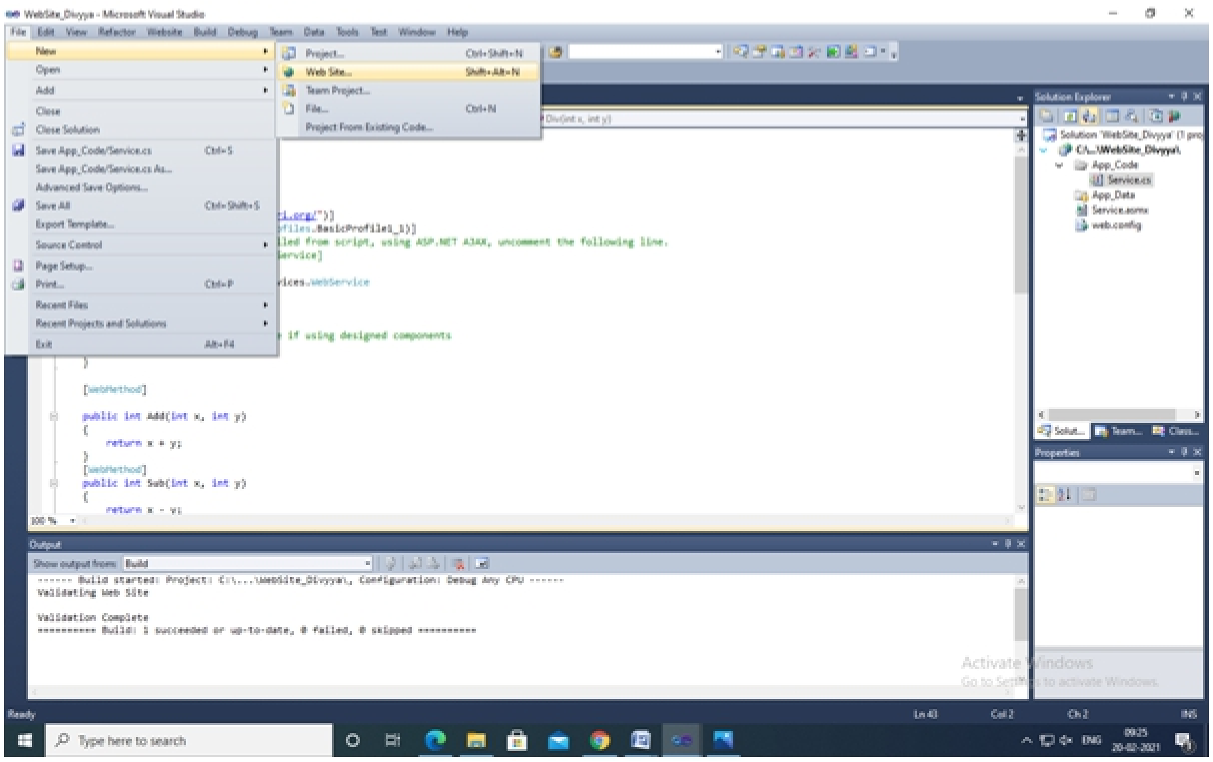
9. Some vendors want to retain their intellectual property rights to certain Web services standards.

A web service can perform almost any kind of task
1. Web Portal- A web portal might obtain top news headlines from an associated press web service. 2. Weather Reporting- It can use as Weather reporting web service to display weather information in your personal website. 3. Stock Quote- It can display latest update of Share market with Stock Quote on your web site. 4. News Headline- You can display latest news update by using News Headline Web Service in your website. 5. You can make your own web service and let others use it. For example, you can make Free SMS Sending Service with footer

with your companies’ advertisement, so whosoever uses this service indirectly advertises your company. You can apply your ideas in N no. of ways to take advantage of it.

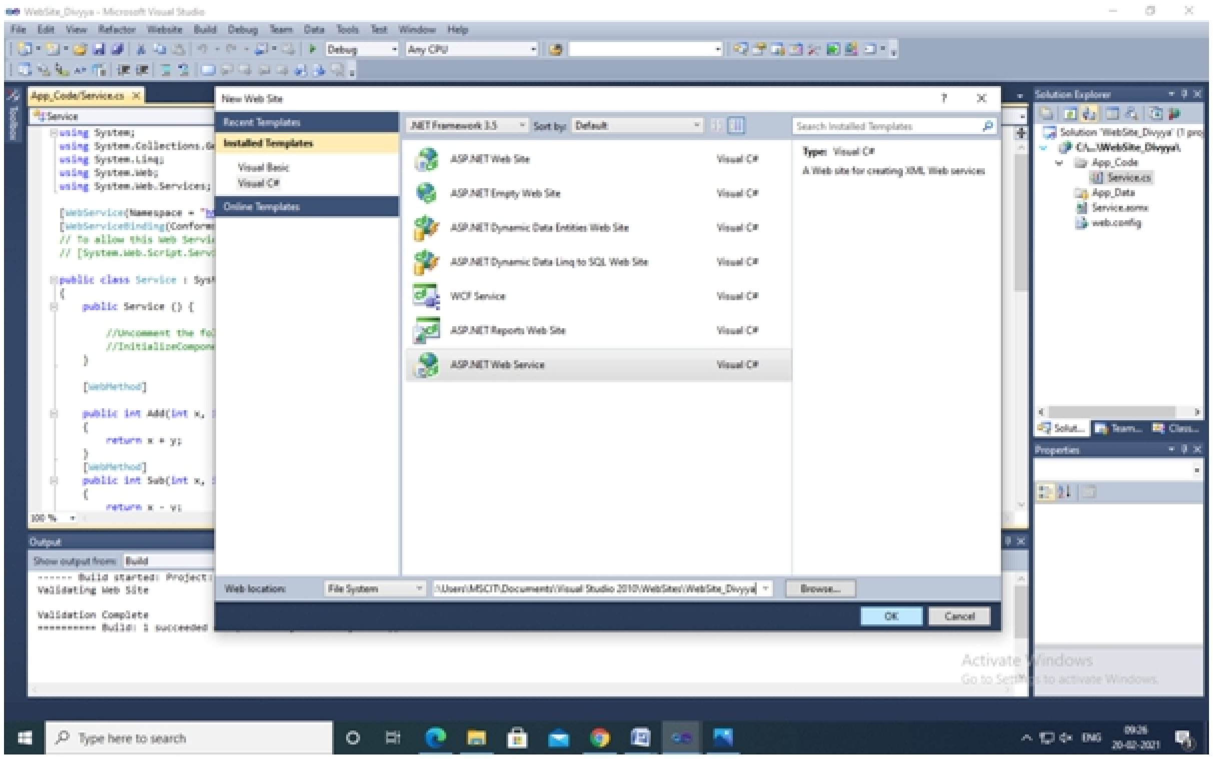
Step 1: Create a ASP.NET Web Service.

Click on File->New->Website->ASP.NET Web Service and name the Web Service.



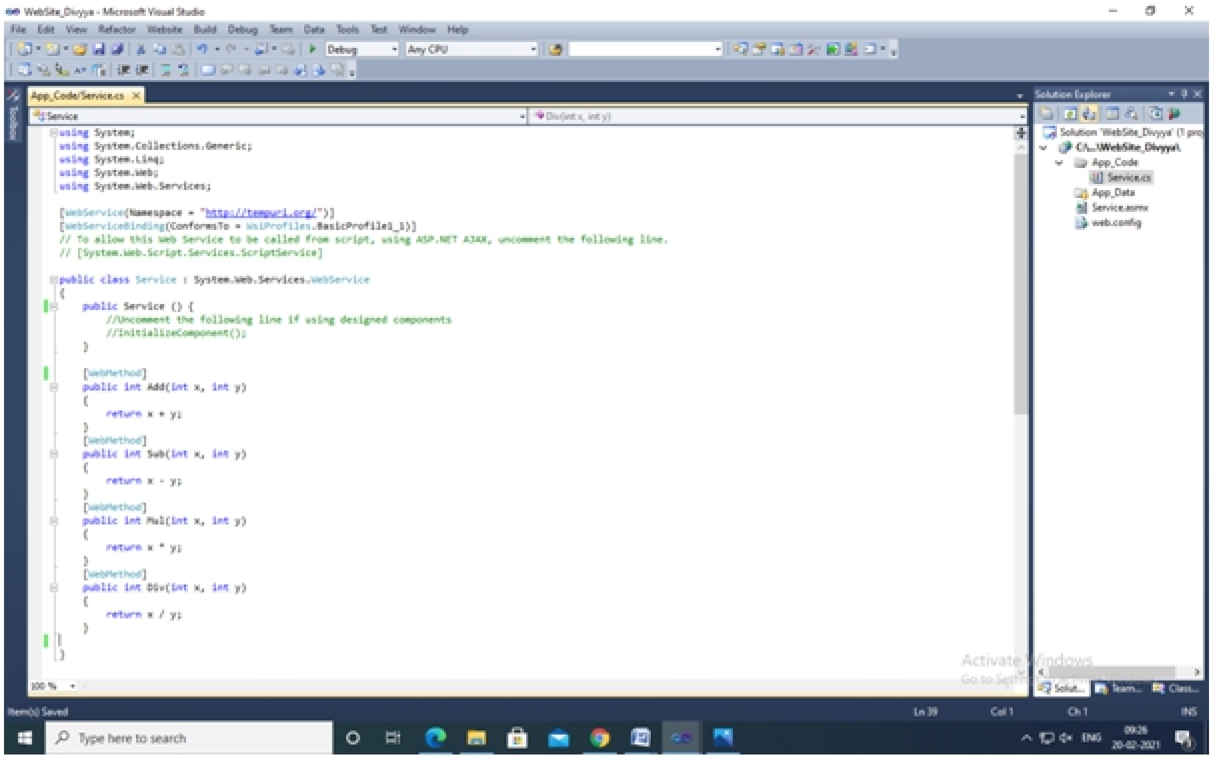
```
i.org/")]
ofiles.BasicProfile1_1)]
lled from script, using ASP.NET AJAX, uncomment the following line.
Service]

vices.WebService

 if using designed components

    }

    [WebMethod]

    public int Add(int x, int y)
    {
        return x + y;
    }
    [WebMethod]
    public int Sub(int x, int y)
    {
        return x - y;
```

```
------ Build started: Project: C:\...\WebSite_Divyya\, Configuration: Debug Any CPU ------
Validating Web Site

Validation Complete
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```


New Web Site
Recent Templates
Installed Templates
Visual Basic
Visual C#
Online Templates
.NET Framework 3.5
Sort by: Default
Search Installed Templates
ASP.NET Web Site — Visual C#
ASP.NET Empty Web Site — Visual C#
ASP.NET Dynamic Data Entities Web Site — Visual C#
ASP.NET Dynamic Data Linq to SQL Web Site — Visual C#
WCF Service — Visual C#
ASP.NET Reports Web Site — Visual C#
ASP.NET Web Service — Visual C#
Type: Visual C#
A Web site for creating XML Web services
Web location: File System
Browse...
OK
Cancel

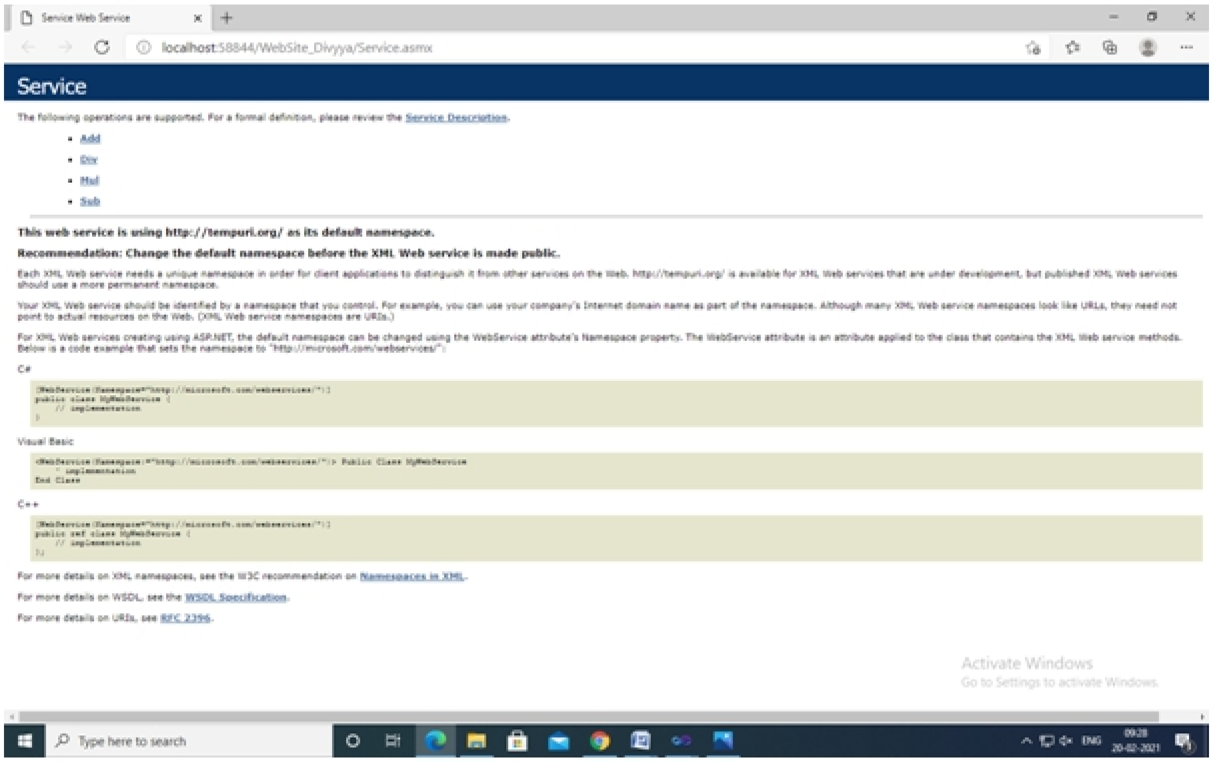
```
\Users\MSCIT\Documents\Visual Studio 2010\WebSites\WebSite_Divyya
```

Step 2: Type the following code:



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

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
// [System.Web.Script.Services.ScriptService]

public class Service : System.Web.Services.WebService
{
    public Service () {
        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    public int Add(int x, int y)
    {
        return x + y;
    }
    [WebMethod]
    public int Sub(int x, int y)
    {
        return x - y;
    }
    [WebMethod]
    public int Mul(int x, int y)
    {
        return x * y;
    }
    [WebMethod]
    public int Div(int x, int y)
    {
        return x / y;
    }
}
}
```

Step 3: Debug the Web Service and Copy the path



localhost:58844/WebSite_Divyya/Service.asmx

Service

The following operations are supported. For a formal definition, please review the Service Description.

- Add
- Div
- Mul
- Sub

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

C++

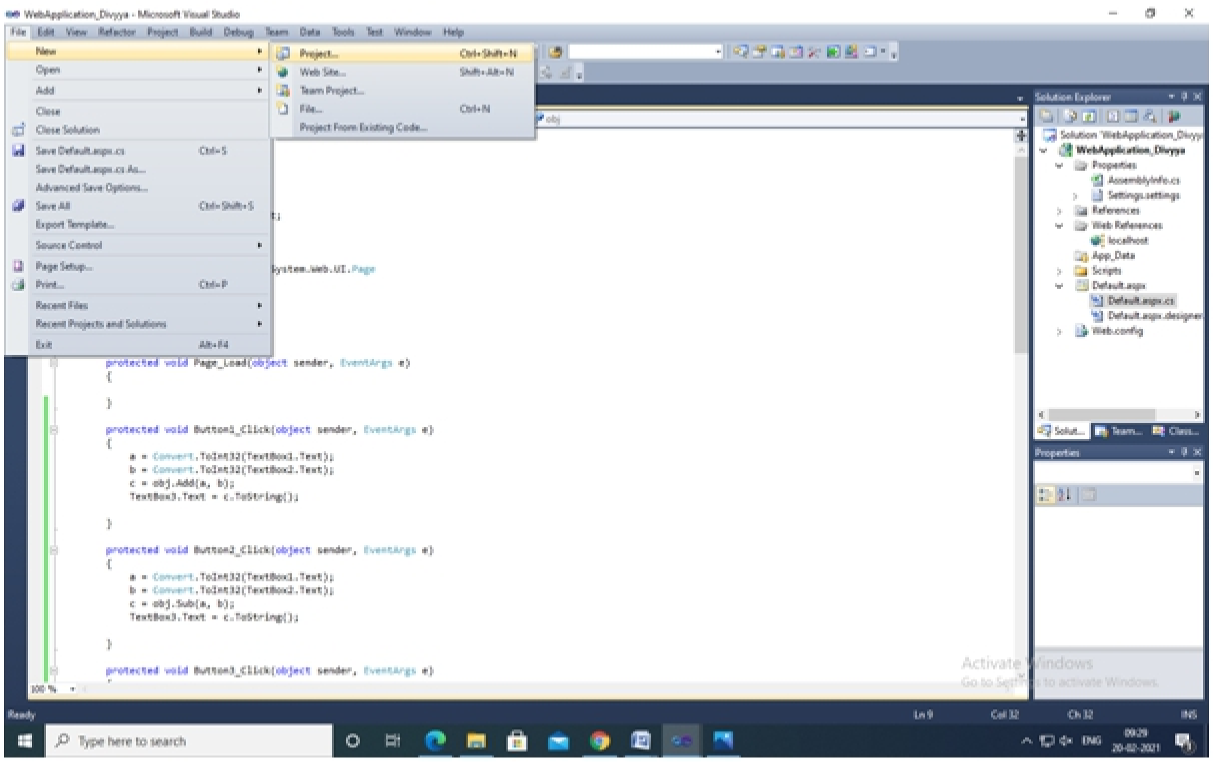
```
[WebService(Namespace="http://microsoft.com/webservices/")]
public ref class MyWebService {
    // implementation
};
```

For more details on XML namespaces, see the W3C recommendation on Namespaces in XML.

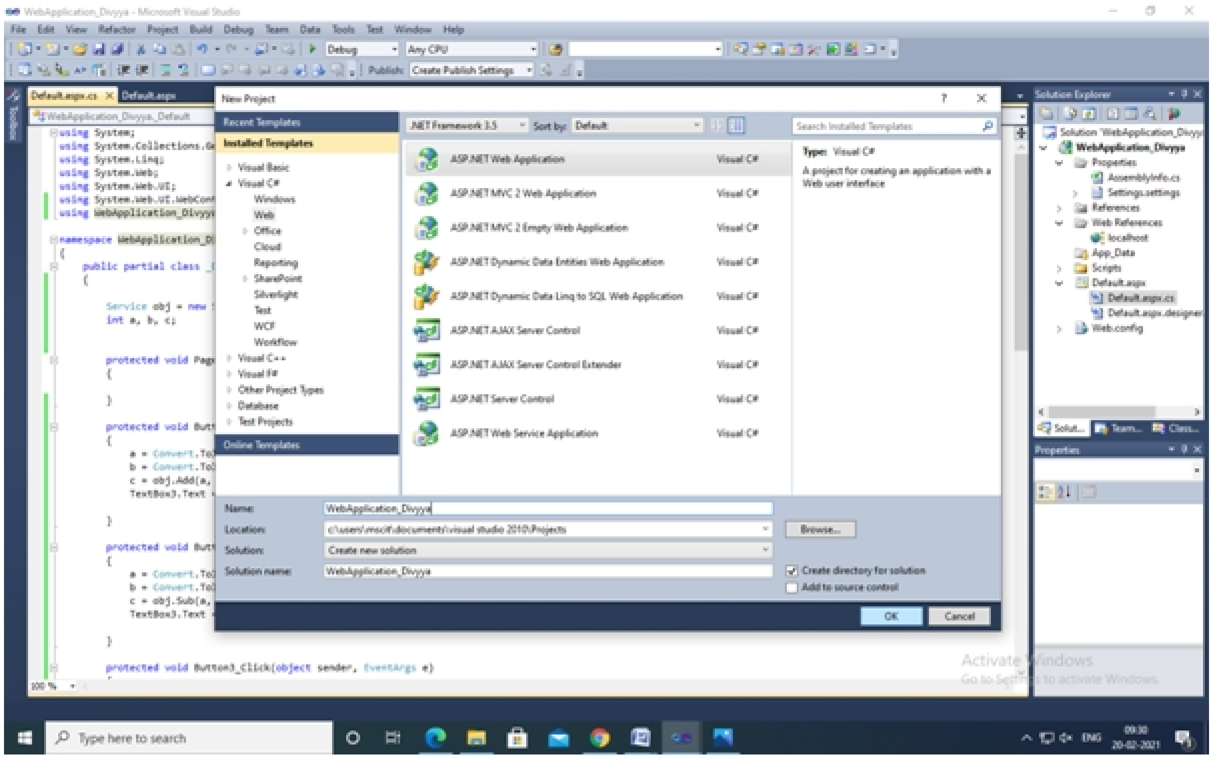
For more details on WSDL, see the WSDL Specification.

For more details on URIs, see RFC 2396.

Step 4: Create a ASP.NET We Application. Click on File->New->Project->Web Application and name the Web Application.



```
System.Web.UI.Page

        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(TextBox1.Text);
            b = Convert.ToInt32(TextBox2.Text);
            c = obj.Add(a, b);
            TextBox3.Text = c.ToString();

        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(TextBox1.Text);
            b = Convert.ToInt32(TextBox2.Text);
            c = obj.Sub(a, b);
            TextBox3.Text = c.ToString();

        }

        protected void Button3_Click(object sender, EventArgs e)
```



WebApplication_Divyya - Microsoft Visual Studio

```
using System;
using System.Collections.G
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebCont
using WebApplication_Divyy

namespace WebApplication_D
{
    public partial class _
    {

        Service obj = new
        int a, b, c;


        protected void Page
        {

        }

        protected void Butt
        {
            a = Convert.To
            b = Convert.To
            c = obj.Add(a,
            TextBox3.Text

        }

        protected void Butt
        {
            a = Convert.To
            b = Convert.To
            c = obj.Sub(a,
            TextBox3.Text

        }

        protected void Button3_Click(object sender, EventArgs e)
```

New Project

Installed Templates: Visual C# › Web

.NET Framework 3.5 | Sort by: Default

ASP.NET Web Application	Visual C#
ASP.NET MVC 2 Web Application	Visual C#
ASP.NET MVC 2 Empty Web Application	Visual C#
ASP.NET Dynamic Data Entities Web Application	Visual C#
ASP.NET Dynamic Data Linq to SQL Web Application	Visual C#
ASP.NET AJAX Server Control	Visual C#
ASP.NET AJAX Server Control Extender	Visual C#
ASP.NET Server Control	Visual C#
ASP.NET Web Service Application	Visual C#

Type: Visual C#

A project for creating an application with a Web user interface

Name: WebApplication_Divyya

Location: c:\users\mscit\documents\visual studio 2010\Projects

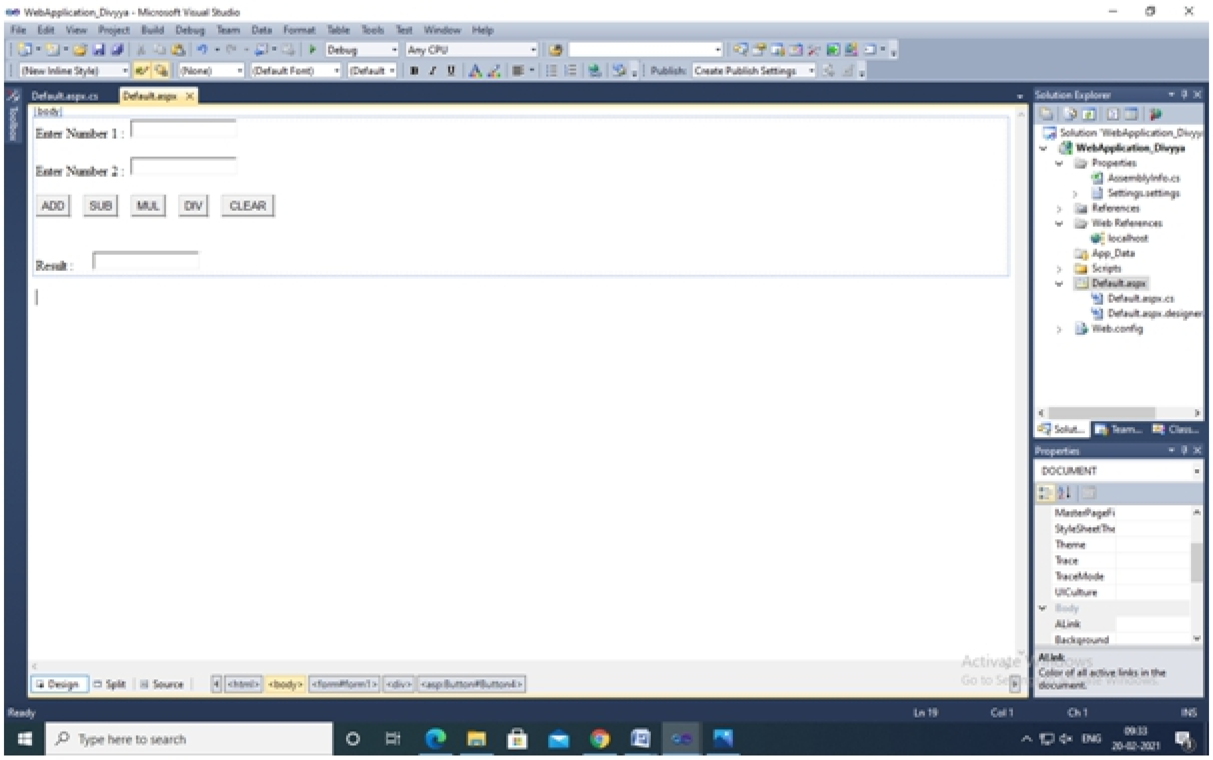
Solution: Create new solution

Solution name: WebApplication_Divyya

[✓] Create directory for solution

[ ] Add to source control

Step 5: Design the simple calculator.



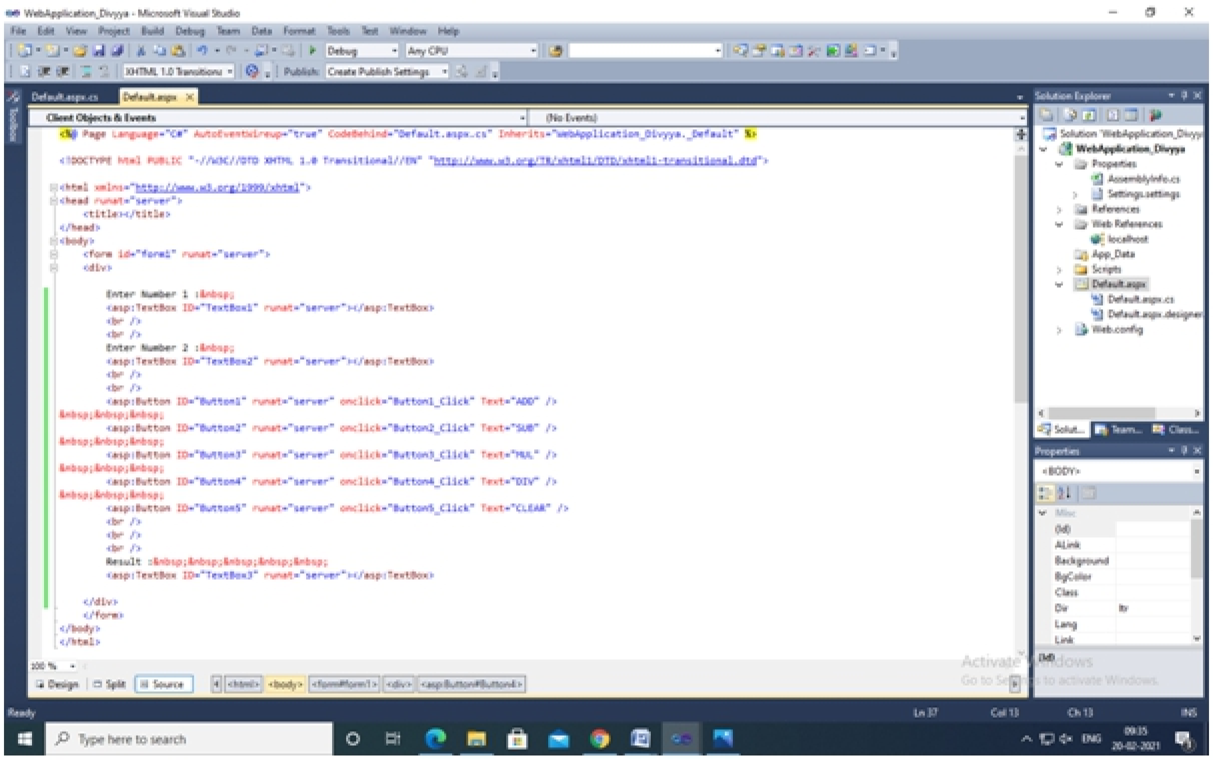
WebApplication_Divyya - Microsoft Visual Studio

Enter Number 1 :

Enter Number 2 :

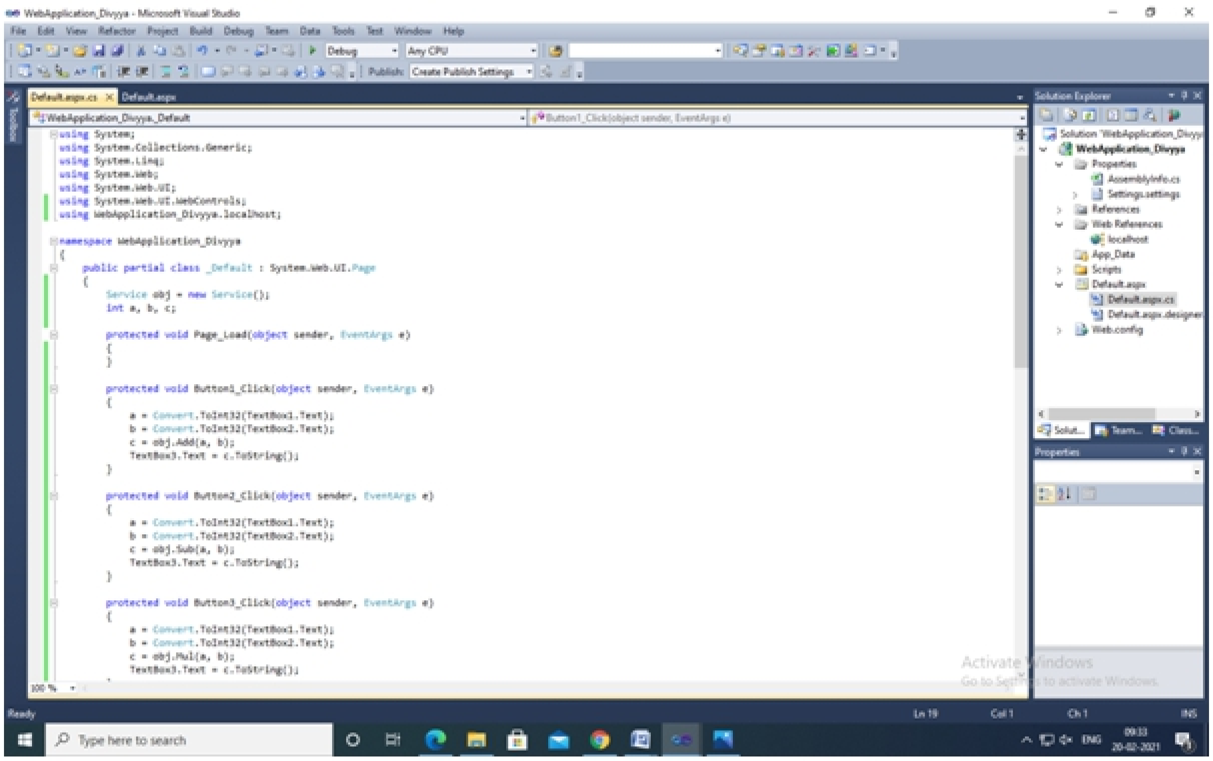
ADD SUB MUL DIV CLEAR

Result :



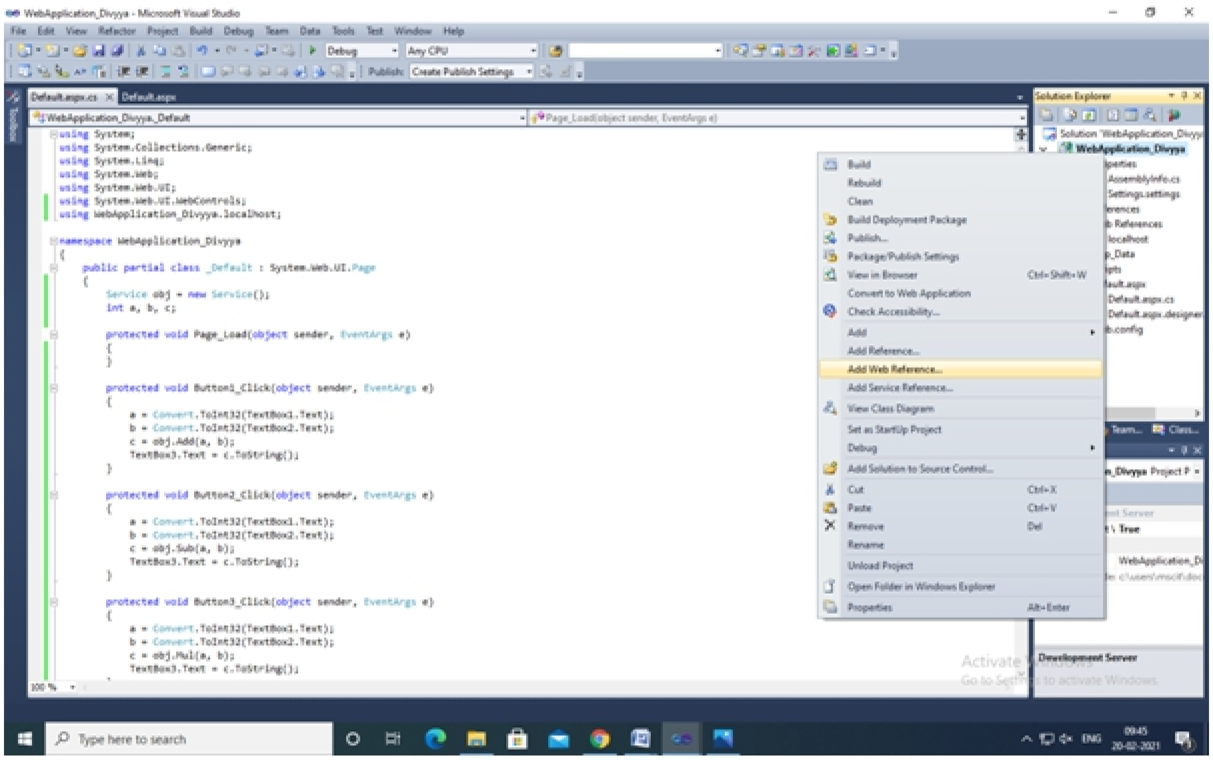
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="WebApplication_Divyya._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>

        Enter Number 1 :&nbsp;
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        Enter Number 2 :&nbsp;
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="ADD" />
&nbsp;&nbsp;&nbsp;
        <asp:Button ID="Button2" runat="server" onclick="Button2_Click" Text="SUB" />
&nbsp;&nbsp;&nbsp;
        <asp:Button ID="Button3" runat="server" onclick="Button3_Click" Text="MUL" />
&nbsp;&nbsp;&nbsp;
        <asp:Button ID="Button4" runat="server" onclick="Button4_Click" Text="DIV" />
&nbsp;&nbsp;&nbsp;
        <asp:Button ID="Button5" runat="server" onclick="Button5_Click" Text="CLEAR" />
        <br />
        <br />
        <br />
        Result :&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;
        <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>

    </div>
    </form>
</body>
</html>
```

Step 6: Type the following code.

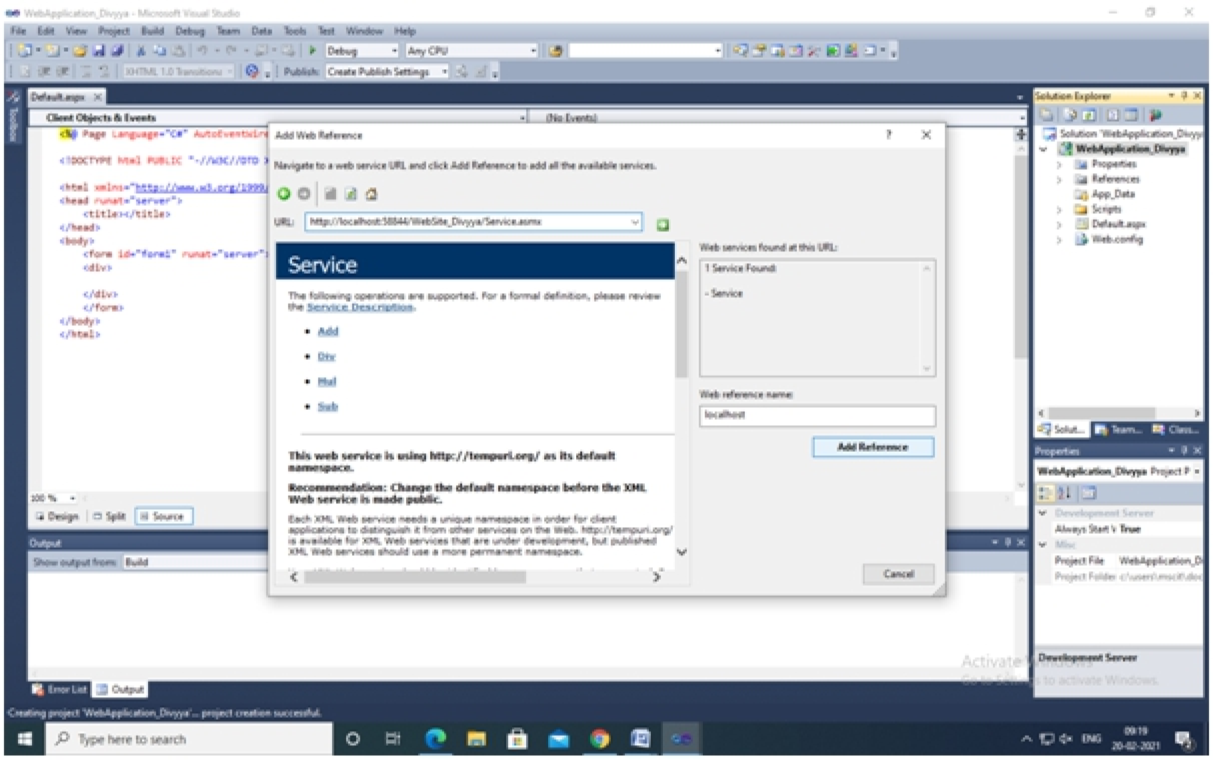


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using WebApplication_Divyya.localhost;

namespace WebApplication_Divyya
{
    public partial class _Default : System.Web.UI.Page
    {
        Service obj = new Service();
        int a, b, c;

        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(TextBox1.Text);
            b = Convert.ToInt32(TextBox2.Text);
            c = obj.Add(a, b);
            TextBox3.Text = c.ToString();
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(TextBox1.Text);
            b = Convert.ToInt32(TextBox2.Text);
            c = obj.Sub(a, b);
            TextBox3.Text = c.ToString();
        }

        protected void Button3_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(TextBox1.Text);
            b = Convert.ToInt32(TextBox2.Text);
            c = obj.Mul(a, b);
            TextBox3.Text = c.ToString();
```

Step 7: Right click on the web application and click ->Add Web reference and copy the path and click on add Reference.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using WebApplication_Divyya.localhost;

namespace WebApplication_Divyya
{
    public partial class _Default : System.Web.UI.Page
    {
        Service obj = new Service();
        int a, b, c;

        protected void Page_Load(object sender, EventArgs e)
        {
        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(TextBox1.Text);
            b = Convert.ToInt32(TextBox2.Text);
            c = obj.Add(a, b);
            TextBox3.Text = c.ToString();
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(TextBox1.Text);
            b = Convert.ToInt32(TextBox2.Text);
            c = obj.Sub(a, b);
            TextBox3.Text = c.ToString();
        }

        protected void Button3_Click(object sender, EventArgs e)
        {
            a = Convert.ToInt32(TextBox1.Text);
            b = Convert.ToInt32(TextBox2.Text);
            c = obj.Mul(a, b);
            TextBox3.Text = c.ToString();
```



Add Web Reference

Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://localhost:58844/WebSite_Divyya/Service.asmx

Service

The following operations are supported. For a formal definition, please review the Service Description.

- Add
- Div
- Mul
- Sub

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Web services found at this URL:

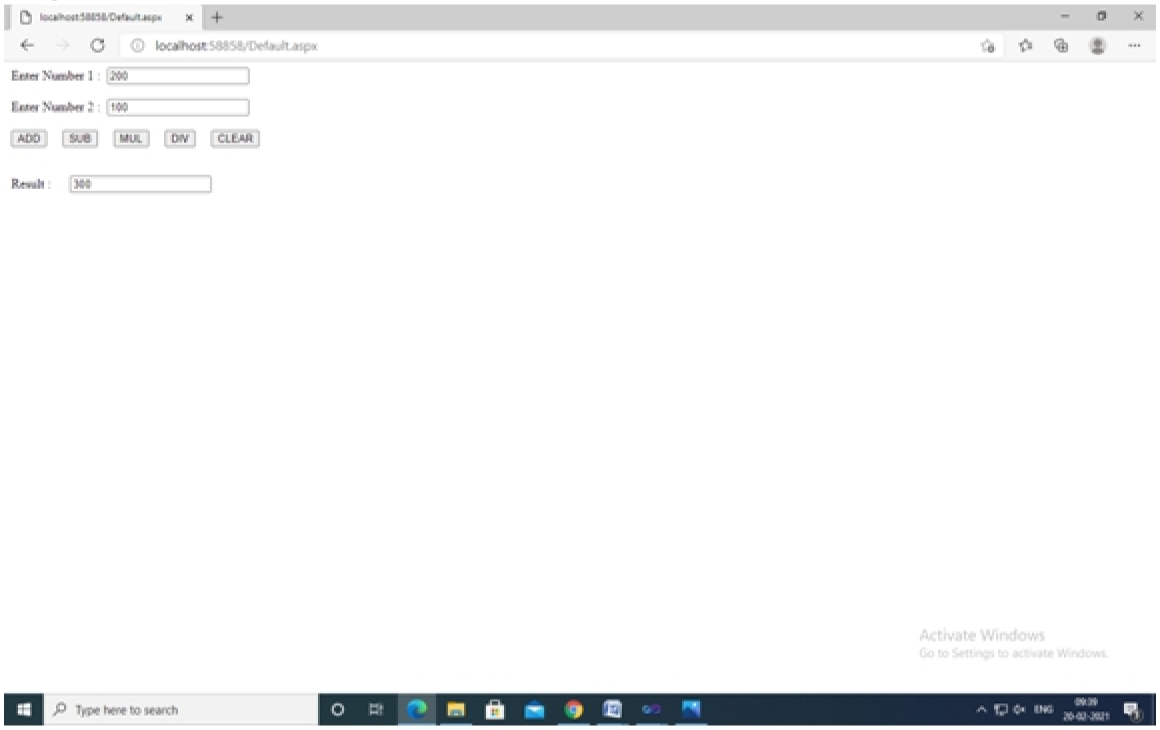
1 Service Found:

- Service

Web reference name: localhost

Step 8: Debug the Web application.

Output:


localhost:58858/Default.aspx
localhost:58858/Default.aspx
Enter Number 1 : 200
Enter Number 2 : 100
ADD
SUB
MUL
DIV
CLEAR
Result : 300
Activate Windows
Go to Settings to activate Windows.
Type here to search
ENG
09:39
20-02-2021

Practical 05

Aim: Write a program to execute any one mutual exclusion algorithm.

<u>Mutual exclusion</u>

Mutual exclusion is a concurrency control property which is introduced to prevent race conditions. It is the requirement that a process cannot enter its critical section while another concurrent process is currently present or executing in its critical section i.e. only one process is allowed to execute the critical section at any given instance of time.

Distributed mutual exclusion algorithms must deal with unpredictable message delays and incomplete knowledge of the system state.

Three basic approaches for distributed mutual exclusion:

1. Token based approach
2. Non-token-based approach
3. Quorum based approach

Token-based approach:

- A unique token is shared among the sites.
- A site is allowed to enter its CS if it possesses the token.
- Mutual exclusion is ensured because the token is unique.
- Example: Suzuki-Kasami's Broadcast Algorithm
  Non-token-based approach:
- Two or more successive rounds of messages are exchanged among the sites to determine which site will enter the CS next.
- Example: Lamport's algorithm, Ricart–Agrawala algorithm ***Quorum based approach:***
- Each site requests permission to execute the CS from a subset of sites (called a quorum).
- Any two quorums contain a common site.
- This common site is responsible to make sure that only one request executes the CS at any time.
- Example: Maekawa's Algorithm

Requirements of Mutual Exclusion Algorithms:

1. **No Deadlock:** Two or more site should not endlessly wait for any message that will never arrive.
2. **No Starvation:** Every site who wants to execute critical section should get an opportunity to execute it in finite time. Any site should not wait indefinitely to execute critical section while other site is repeatedly executing critical section

3. **Fairness:** Each site should get a fair chance to execute critical section. Any request to execute critical section must be executed in the order they are made i.e Critical section execution requests should be executed in the order of their arrival in the system.
4. **Fault Tolerance:** In case of failure, it should be able to recognize it by itself in order to continue functioning without any disruption.

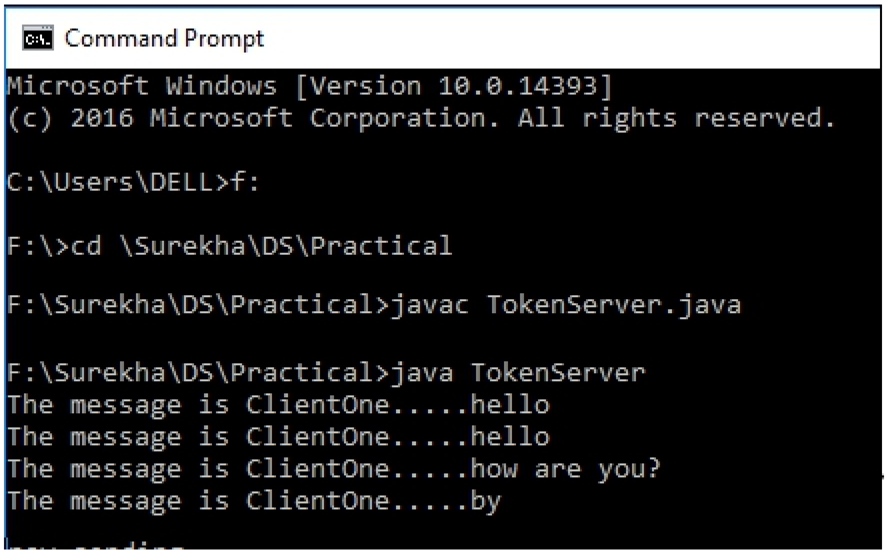
5A: Write a program to execute any one mutual exclusion algorithm.

Program –

TokenServer.java

```
import java.net.*;
public class TokenServer
{
   public static void main(String agrs[])throws Exception
   {
      while(true)
      {
         Server sr=new Server();
         sr.recPort(8000);
         sr.recData();
      }
   }
}
class Server
{
   boolean hasToken=false;
   boolean sendData=false;
   int recport;
   void recPort(int recport)
   {
      this.recport=recport;
   }
   void recData()throws Exception
   {
      byte buff[]=new byte[256];
      DatagramSocket ds;
      DatagramPacket dp;
      String str;
      ds=new DatagramSocket(recport);
      dp=new DatagramPacket(buff,buff.length);
      ds.receive(dp);
                  ds.close();
str=new String(dp.getData(),0,dp.getLength());
System.out.println("The message is "+str);
}
}
```

Output:



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac TokenServer.java

F:\Surekha\DS\Practical>java TokenServer
The message is ClientOne.....hello
The message is ClientOne.....hello
The message is ClientOne.....how are you?
The message is ClientOne.....by
```

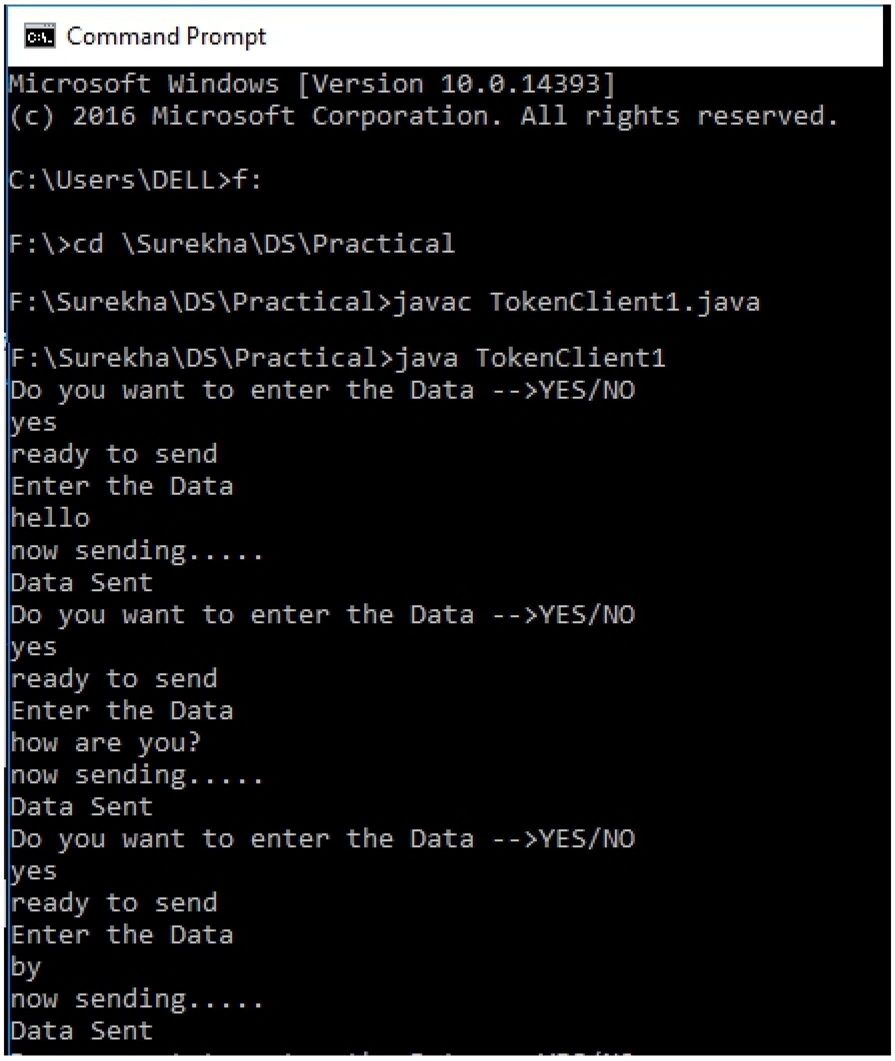
TokenClient1.java

```
import java.io.*;
import java.net.*;
public class TokenClient1
{
public static void main(String arg[]) throws Exception
{
InetAddress lclhost;
BufferedReader br;
String str="";
TokenClient12 tkcl,tkser;
boolean hasToken;
boolean setSendData;
while(true)
{
lclhost=InetAddress.getLocalHost();
tkcl = new TokenClient12(lclhost);
tkser = new TokenClient12(lclhost);
tkcl.setSendPort(9004);
tkcl.setRecPort(8002);
lclhost=InetAddress.getLocalHost();
tkser.setSendPort(9000);
if(tkcl.hasToken == true)
{
System.out.println("Do you want to enter the Data -->YES/NO");
br=new BufferedReader(new InputStreamReader(System.in));
str=br.readLine();
if(str.equalsIgnoreCase("yes"))
{
System.out.println("ready to send");
tkser.setSendData = true;
tkser.sendData();
tkser.setSendData = false;
}
else if(str.equalsIgnoreCase("no"))
{
tkcl.sendData();
tkcl.recData();
}
}
else
{
System.out.println("ENTERING RECEIVING MODE...");
tkcl.recData();
}
}
}
}
class TokenClient12
{
InetAddress lclhost;
```

```
int sendport,recport;
boolean hasToken = true;
boolean setSendData = false;
TokenClient12 tkcl,tkser;
TokenClient12(InetAddress lclhost)
{
this.lclhost = lclhost;
}
void setSendPort(int sendport)
{
this.sendport = sendport;
}
void setRecPort(int recport)
{
this.recport = recport;
}
void sendData() throws Exception
{
BufferedReader br;
String str="Token";
DatagramSocket ds;
DatagramPacket dp;
if(setSendData == true)
{
System.out.println("Enter the Data");
br=new BufferedReader(new InputStreamReader(System.in));
str = "ClientOne....." + br.readLine();
System.out.println("now sending.....");
System.out.println("Data Sent");
}
ds = new DatagramSocket(sendport);
dp = new DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);
ds.send(dp);
ds.close();
setSendData = false;
hasToken = false;
}
void recData()throws Exception
{
String msgstr;
byte buffer[] = new byte[256];
DatagramSocket ds;
DatagramPacket dp;
ds = new DatagramSocket(recport);
dp = new DatagramPacket(buffer,buffer.length);
ds.receive(dp);
ds.close();
msgstr = new String(dp.getData(),0,dp.getLength());
System.out.println("The data is "+msgstr);
if(msgstr.equals("Token"))
{
```

```
hasToken = true;
}
}
}
```

Output:



Command Prompt

```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\DELL>f:

F:\>cd \Surekha\DS\Practical

F:\Surekha\DS\Practical>javac TokenClient1.java

F:\Surekha\DS\Practical>java TokenClient1
Do you want to enter the Data -->YES/NO
yes
ready to send
Enter the Data
hello
now sending.....
Data Sent
Do you want to enter the Data -->YES/NO
yes
ready to send
Enter the Data
how are you?
now sending.....
Data Sent
Do you want to enter the Data -->YES/NO
yes
ready to send
Enter the Data
by
now sending.....
Data Sent
```

<u>TokenClient2.java</u>

```
import java.io.*;
import java.net.*;
public class TokenClient2
{
static boolean setSendData ;
static boolean hasToken ;
public static void main(String arg[]) throws Exception
{
InetAddress lclhost;
BufferedReader br;
String str1;
TokenClient21 tkcl;
TokenClient21 ser;
while(true)
{
lclhost=InetAddress.getLocalHost();
tkcl = new TokenClient21(lclhost);
tkcl.setSendPort(9004);
tkcl.setRecPort(8002);
lclhost=InetAddress.getLocalHost();
ser = new TokenClient21(lclhost);
ser.setSendPort(9000);
if(hasToken == true)
{
System.out.println("Do you want to enter the Data -->YES/NO");
br=new BufferedReader(new InputStreamReader(System.in));
str1=br.readLine();
if(str1.equalsIgnoreCase("yes"))
{
System.out.println("ready to send");
ser.setSendData = true;
ser.sendData();
ser.setSendData = false;
}
else if(str1.equalsIgnoreCase("no"))
{
tkcl.sendData();
tkcl.recData();
}
}
else
{
System.out.println("entering recieving mode");
tkcl.recData();
hasToken=true;
}
}
}
}
class TokenClient21
{
InetAddress lclhost;
```

```
int sendport,recport;
boolean setSendData = false;
boolean hasToken = false;
TokenClient21 tkcl;
TokenClient21 ser;
TokenClient21(InetAddress lclhost)
{
this.lclhost = lclhost;
}
void setSendPort(int sendport)
{
this.sendport = sendport;
}
void setRecPort(int recport)
{
this.recport = recport;
}
void sendData() throws Exception
{
BufferedReader br;
String str = "Token";
DatagramSocket ds;
DatagramPacket dp;
if(setSendData == true)
{
System.out.println("Enter the Data");
br=new BufferedReader(new InputStreamReader(System.in));
str = "ClientTwo....." + br.readLine();
System.out.println("now sending.....");
System.out.println("Data Sent");
}
ds = new DatagramSocket(sendport);
dp = new DatagramPacket(str.getBytes(),str.length(),lclhost,sendport-1000);
ds.send(dp);
ds.close();
setSendData = false;
hasToken = false;
}
void recData()throws Exception
{
String msgstr;
byte buffer[] = new byte[256];
DatagramSocket ds;
DatagramPacket dp;
ds = new DatagramSocket(recport);
dp = new DatagramPacket(buffer,buffer.length);
ds.receive(dp);
ds.close();
msgstr = new String(dp.getData(),0,dp.getLength());
System.out.println("The data is "+msgstr);
if(msgstr.equals("Token"))
{
hasToken = true;
}
}
}
```

Practical 06

Aim: Write a program to implement any one election algorithm.

Election algorithms

Many distributed algorithms require the election of a special coordinator process; one that has a special role, or initiates something, or monitors something. Often it doesn't matter who the special process is, but one and only one must be elected, and it can't be known in advance who it will be. On a low-level you can think about the monitor in a token ring as an example.
The assumptions of these algorithms are that every process can be uniquely identified (by IP address, for example), and that each process can find out the id of the other processes. What the processes don't know is which processes are up and which are down at any given point in time.

Bully algorithm

Garcia-Molina, 1982. The process with the highest identity always becomes the coordinator. When a process P sees that the coordinator is no longer responding to requests it initiates an election by sending ELECTION messages to all processes whose id is higher than its own. If no one responds to the messages then P is the new coordinator. If one of the higher-ups responds, it takes over and P doesn't have to worry anymore.
When a process receives an ELECTION message it sends a response back saying OK. It then holds its own election (unless it is already holding one). Eventually there is only one process that has not given up and that is the new coordinator. It is also the one with the highest number currently running. When the election is done the new coordinator sends a COORDINATOR message to everyone informing them of the change.
If a process which was down comes back up, it immediately holds an election. If this process had previously been the coordinator it will take this role back from whoever is doing it currently (hence the name of the algorithm).

Ring algorithm

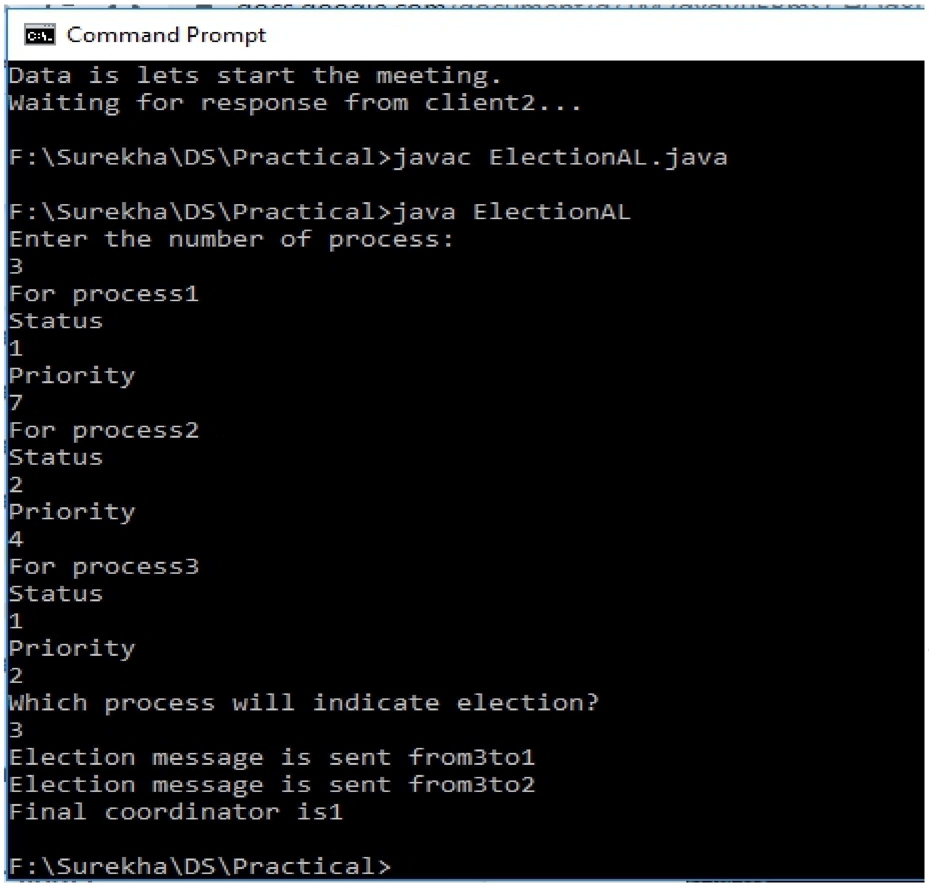
Assumes that processes are logically ordered in some fashion, and that each process knows the order and who is coordinator. No token is involved. When a process notices that the coordinator is not responding it sends an ELECTION message with its own id to its downstream neighbor. If that neighbor doesn't respond it sends it to its neighbor's neighbor, etc. Each station that receives the ELECTION message adds its own id to the list. When the message circulates back to the originator it selects the highest id in the list and sends a COORDINATOR message announcing the new coordinator. This message circulates once and is removed by the originator. If two elections are held simultaneously (say because two different processes notice simultaneously that the coordinator is dead) then each comes up with the same list and elects the same coordinator. Some time is wasted, but nothing is really hurt by this.

Program:

ElectionAL.java

```
import java.io.*;
import java.util.Scanner;
class ElectionAL
{
   static int n;
   static int pro[]= new int[100];
   static int sta[]= new int[100];
   static int co;

   public static void main(String args[])throws IOException
   {
          System.out.println("Enter the number of process:");
          Scanner in = new Scanner(System.in);
          n = in.nextInt();
          int i;
          for(i=0;i<n;i++)
          {
                   System.out.println("For process"+(i+1)+"");
                   System.out.println("Status");
                   sta[i] = in.nextInt();
                   System.out.println("Priority");
                   pro[i] = in.nextInt();
          }
          System.out.println("Which process will indicate election?");
          int ele = in.nextInt();
          elect(ele);
          System.out.println("Final coordinator is"+co);
   }
   static void elect(int ele)
   {
          ele = ele-1;
          co = ele+1;
          for(int i=0;i<n;i++)
          {
                   if(pro[ele]<pro[i])
                   {
                            System.out.println("Election message is sent from"+(ele+1)+"to"+(i+1));
                            if(sta[i]==1)
                                     elect(i+1);
                   }
          }
   }
}
```

output:


Command Prompt

```
Data is lets start the meeting.
Waiting for response from client2...

F:\Surekha\DS\Practical>javac ElectionAL.java

F:\Surekha\DS\Practical>java ElectionAL
Enter the number of process:
3
For process1
Status
1
Priority
7
For process2
Status
2
Priority
4
For process3
Status
1
Priority
2
Which process will indicate election?
3
Election message is sent from3to1
Election message is sent from3to2
Final coordinator is1

F:\Surekha\DS\Practical>
```

Practical 07

Aim: Show the implementation of any one clock synchronization algorithm.

Clock synchronization algorithm:

Clock synchronization is a method of synchronizing clock values of any two nodes in a distributed system with the use of external reference clock or internal clock value of the node. During the synchronization, many factors effect on a network. As discussed above, these factors need to be considered before correcting actual clock value. Based on the approach, clock synchronization algorithms are divided as Centralized Clock Synchronization and Distributed Clock Synchronization Algorithm.
Distributed algorithm:
• There is particular time server.
• The processor periodically reach an agreement on the clock value by averaging the time of neighbor clock and its local clock.
• This can be used if no UTC receiver exist (no external synchronization is needed). Only internal synchronization is performed.
• Processes can run different machine and no global clock to judge which event happened first.

Aim: Write the implementation of any one clock synchronization algorithm

Program:

SCServer.java

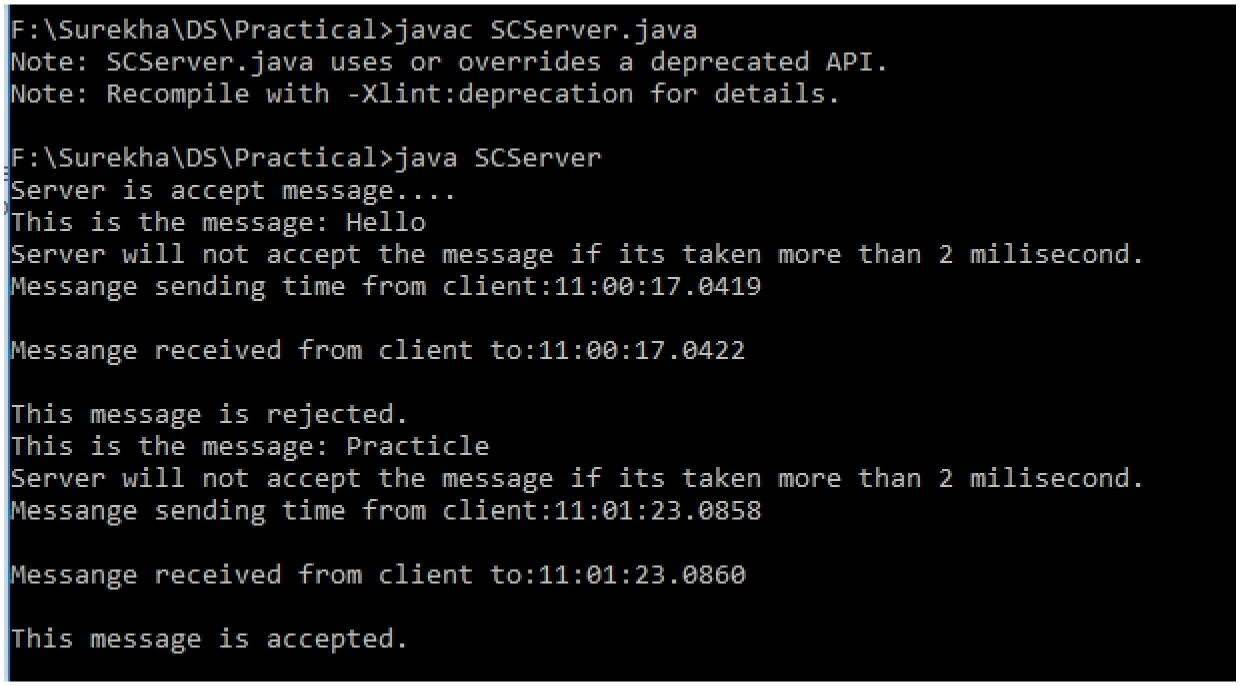
```
import java.net.*;
import java.io.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;

public class SCServer {

    public static void main(String[] args) {

        final int PORT = 8001;

        try {

            ServerSocket serverSocket = new ServerSocket(PORT);
            System.out.println("Server is accepting message at PORT " + PORT);

            Socket socket = serverSocket.accept();

            DataInputStream inputStream = new DataInputStream(socket.getInputStream());

            String receieve;

            DateFormat dateFormat = new SimpleDateFormat("hh:mm:ss.SSSS");

            while ((receieve = inputStream.readLine()) != null) {
                String[] message = receieve.split(",");
                if (message[0].toLowerCase().equals("stop"))
                    break;

                Date date = new Date();

                Long clientTime = Long.parseLong(message[1]);
                Long timeMilli = date.getTime();

                Long requiredTime = timeMilli - clientTime;

                System.out.println("This is the message: " + message[0]);

                System.out.println("Server will not accept the message if its taken more than 2
milisecond.");

                String formattedClientTime = dateFormat.format(clientTime);
                if (clientTime.equals(timeMilli)) {
                    String strDate = dateFormat.format(timeMilli);
                    System.out.println("Message sending time and receving time is same:" + strDate);
                } else if (requiredTime > 2) {
                    System.out.println("Message sending time from client:" + formattedClientTime +
"\n");
                    String strDate = dateFormat.format(timeMilli);
                    System.out.println("Message received from client to:" + strDate + "\n");
                    System.out.println("This message is rejected.");
                } else {
                    System.out.println("Message sending time from client:" + formattedClientTime +
"\n");
                    String strDate = dateFormat.format(timeMilli);
                    System.out.println("Message received from client to:" + strDate + "\n");
                    System.out.println("This message is accepted.");
                }
            }
            inputStream.close();

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

    }
}
```

SCClient.java

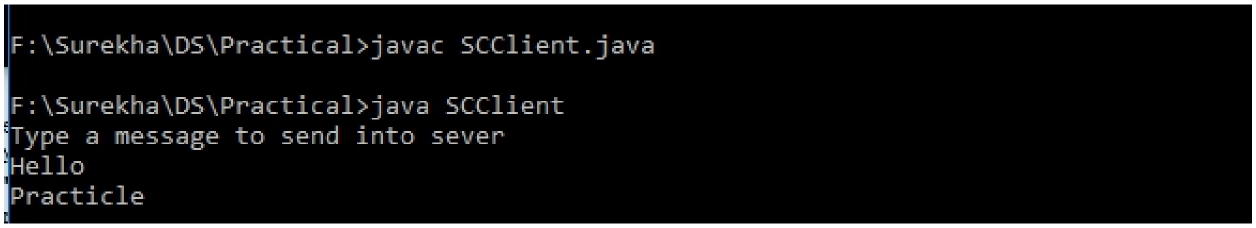
```
import java.net.*;
import java.io.*;
import java.util.Date;
```

```
public class SCClient {
    public static void main(String[] args) {
        final int PORT = 8001;
        final String HOST = "LocalHost";
        try {
            Socket socket = new Socket(HOST, PORT);
            BufferedReader inputBuffer = new BufferedReader(new InputStreamReader(System.in));
            DataOutputStream outputStream = new DataOutputStream(socket.getOutputStream());
            String send;
            System.out.println("Type a message to send into server");
            while ((send = inputBuffer.readLine()) != null) {
                Date date = new Date();
                long timeMilli = date.getTime();
                String t = String.valueOf(timeMilli);
                outputStream.writeBytes(send + "," + t + "\n");
                if (send.toLowerCase().equals("stop"))
                    break;
            }
            inputBuffer.close();
            outputStream.close();
            socket.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output -



```
F:\Surekha\DS\Practical>javac SCServer.java
Note: SCServer.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

F:\Surekha\DS\Practical>java SCServer
Server is accept message....
This is the message: Hello
Server will not accept the message if its taken more than 2 milisecond.
Messange sending time from client:11:00:17.0419

Messange received from client to:11:00:17.0422

This message is rejected.
This is the message: Practicle
Server will not accept the message if its taken more than 2 milisecond.
Messange sending time from client:11:01:23.0858

Messange received from client to:11:01:23.0860

This message is accepted.
```


```
F:\Surekha\DS\Practical>javac SCClient.java

F:\Surekha\DS\Practical>java SCClient
Type a message to send into sever
Hello
Practicle
```

Practical 08

Aim: Write a program to implement two phase commit protocol.

The two-phase commit protocol is an algorithm which lets all sites in a distributed system agree to commit or rollback a transaction based upon consensus of all participating sites. The two-phase commit strategy is designed to ensure that either all the databases are updated or none of them, so that the databases remain synchronized.

In two phase commit protocol, there is one node which acts as a **Coordinator** and all other participating nodes are known as **Cohorts.**

The two-phase commit protocol involves two phases. The first phase is "Commit Request" phase, and the second phase is "Commit" phase.

Commit Request Phase

To commit the transaction, the coordinator sends a request asking for "ready for commit?" to each cohorts.

The coordinator waits until it has received a reply from all cohorts to “vote” on the request.

Each cohort votes by sending a message back to the coordinator, as follows:

- It votes YES if it is prepared to commit
- It may vote NO for any reason, usually because it cannot prepare the transaction due to a local failure.
- It may delay voting indefinitely, for example, because its system was busy with other work because it failed.
  Commit Phase:

If the coordinator receives YES response from all cohorts, it decides to commit. The transaction is now officially committed. Otherwise, it either receives a NO response or gives up waiting for some cohorts, so it decides to abort.

The coordinator sends its decision to all participants (i.e. COMMIT or ABORT).Even if one cohort had voted to abort the transaction then it send ABORT message.

Cohorts acknowledge receipt of commit or abort by replying DONE.

Program -

TPCServer

```
import java.io.*;

import java.net.*;


public class TPCServer{

  public static void main(String a[]) throws Exception{

        BufferedReader br;

        InetAddress lclhost;
```

```
lclhost=InetAddress.getLocalHost();

Server ser=new Server(lclhost);

System.out.println("Server in sending mode....");

//Sending data to client1

ser.setSendPort(9000);  //recport=8000

ser.setRecPort(8001);  //sendport=9001

System.out.println("Send request data to client1....");

br=new BufferedReader(new InputStreamReader(System.in));

String s=br.readLine();

System.out.println("Data is " +s);

ser.sendData();

System.out.println("Waiting for response from client1.....");

ser.recData();

//Sending data to client 2

ser.setSendPort(9002);  //recport=8002

ser.setRecPort(8003);  //senport=9003

System.out.println("Send request data to client2...");

br=new BufferedReader(new InputStreamReader(System.in));

String s1=br.readLine();

System.out.println("Data is " +s1);

ser.sendData();

System.out.println("Waiting for response from client2...");

ser.recData();

//Sending the final result to client 1
```

```
        ser.setSendPort(9000);
         ser.sendData();
         //Sending the final result to client 2
         ser.setSendPort(9002);
         ser.sendData();
            }
}
class Server{
InetAddress lclhost;
int sendPort,recPort;
int ssend=0;
int scounter=0;
Server(InetAddress lclhost)
{
      this.lclhost=lclhost;
}
public void setSendPort(int sendPort)
{
      this.sendPort=sendPort;
}
public void setRecPort(int recPort)
{
       this.recPort=recPort;
}
```

```
public void sendData() throws Exception

{

DatagramSocket ds;

DatagramPacket dp;

String data="";

if(scounter<2 && ssend<2)

{

            data="VOTE_REQUEST";

}

if(scounter<2 && ssend>1)

{

 data="GLOBAL_ABORT";

 data=data+"TRANSACTION ABORTED";

}

if(scounter==2 && ssend>1){

data="GLOBAL_COMMIT";

data=data+"TRANSACTION COMMITTED";

}

ds=new DatagramSocket(sendPort);

dp=new DatagramPacket(data.getBytes(),data.length(), lclhost, sendPort-1000);

ds.send(dp);

ds.close();

ssend++;

}
```

```
public void recData() throws Exception

{

byte buf[]=new byte[256];

DatagramPacket dp=null;

DatagramSocket ds = null;

String msgStr="";

try{

ds=new DatagramSocket(recPort);

dp=new DatagramPacket(buf,buf.length);

ds.receive(dp);

ds.close();

}

catch(Exception e){

e.printStackTrace();

}

msgStr=new String(dp.getData(),0,dp.getLength());

System.out.println("String="+msgStr);

if(msgStr.equalsIgnoreCase("VOTE_COMMIT"))

{

scounter++;

}

System.out.println("Counter value ="+scounter + "n Send value = "+ssend);

}

};
```

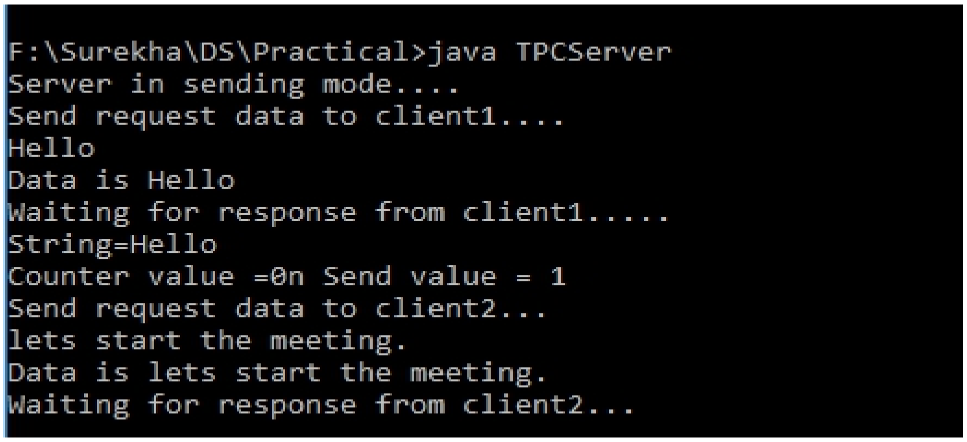
TPCClient1

```
import java.io.*;

import java.net.*;

public class TPCClient1

{

  public static void main(String a[])throws Exception

  {

            InetAddress lclhost;

            lclhost=InetAddress.getLocalHost();

            Client clnt=new Client(lclhost);

            clnt.setSendPort(9001);

            clnt.setRecPort(8000);

            clnt.recData();

            clnt.sendData();

            clnt.recData();

  }

}

class Client{

  InetAddress lclhost;

   int sendPort,recPort;

   Client(InetAddress lclhost)

   {

            this.lclhost=lclhost;

   }
```

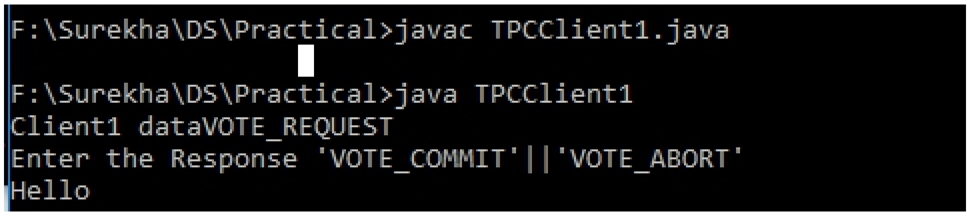
```
        public void setSendPort(int sendPort)
        {
        this.sendPort=sendPort;
        }

        public void setRecPort(int recPort)
        {
        this.recPort=recPort;
        }
        public void sendData()throws Exception
        {
        BufferedReader br;
        DatagramSocket ds;
        DatagramPacket dp;
        String data="";
    System.out.println("Enter the Response 'VOTE_COMMIT'||'VOTE_ABORT'");
        br=new BufferedReader(new InputStreamReader(System.in));
        data = br.readLine();
        System.out.println("Data is"+data);
        ds=new DatagramSocket(sendPort);
        dp=new    DatagramPacket(data.getBytes(),data.length(),lclhost,sendPort-1000);
        ds.send(dp);
        ds.close();
        }
```

```
  public void recData()throws Exception

  {

          byte buf[]=new byte[256];

          DatagramPacket dp;

          DatagramSocket ds;

          ds=new DatagramSocket(recPort);

          dp=new DatagramPacket(buf,buf.length);

          ds.receive(dp);

          ds.close();

          String msgStr=new

          String(dp.getData(),0,dp.getLength());

          System.out.println("Client1 data"+msgStr);

  }

};
```

Output:



```
F:\Surekha\DS\Practical>java TPCServer
Server in sending mode....
Send request data to client1....
Hello
Data is Hello
Waiting for response from client1.....
String=Hello
Counter value =0n Send value = 1
Send request data to client2...
lets start the meeting.
Data is lets start the meeting.
Waiting for response from client2...
```



```
F:\Surekha\DS\Practical>javac TPCClient1.java

F:\Surekha\DS\Practical>java TPCClient1
Client1 dataVOTE_REQUEST
Enter the Response 'VOTE_COMMIT'||'VOTE_ABORT'
Hello
```