



**RAMNIRANJAN JHUNJHUNWALA COLLEGE**  
**GHATKOPAR (W), MUMBAI - 400 086**

**DEPARTMENT OF INFORMATION TECHNOLOGY**  
**2020 - 2021**

**M.Sc.( I.T.) SEM I**  
**Distributed System**

**Name: Shweta Yadav**  
**Roll No.: 21**



Hindi Vidya Prachar Samiti's

**RAMNIRANJAN  
JHUNJHUNWALA COLLEGE  
(AUTONOMOUS)**



Opposite Ghatkopar Railway Station, Ghatkopar West, Mumbai-400086

## CERTIFICATE

This is to certify that Miss. **Shweta Omprakash Yadav** with Roll No. **21** has successfully completed the necessary course of experiments in the subject of **Distributed Systems** 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

---

External Examiner

---

Head of Department

---

College Seal



**INDEX:**

<b>Prac no.</b>	<b>Aim</b>	<b>Date</b>
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 -1**

**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.**

### **ServerPrimeNum.java**

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

public class ServerPrimeNum {
    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());
    }
}
}

```

### **ClientPrimeNum.java**

```

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

public class ClientPrimeNum {
    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);

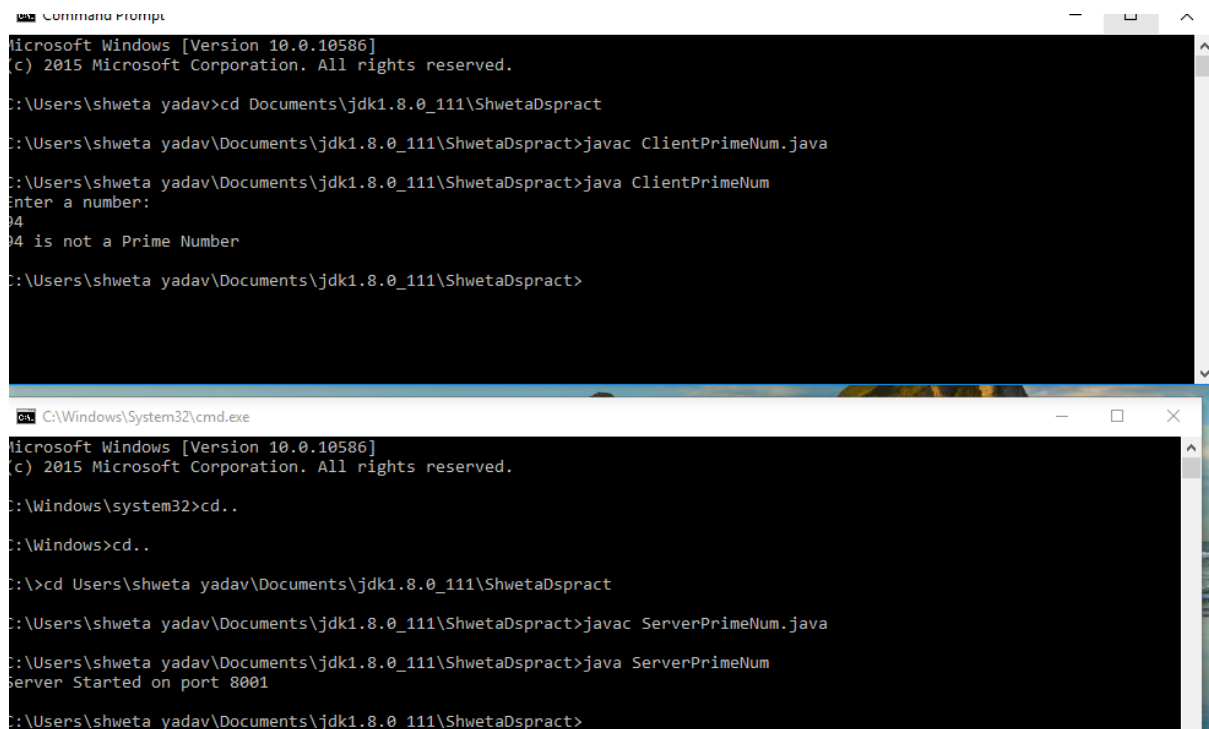
```

```

        DataInputStream in = new DataInputStream(s.getInputStream());
        System.out.println(in.readUTF());
        s.close();
    } catch (Exception e) {
        System.out.println(e.toString());
    }
}
}
}

```

## Output :



```

C:\Users\shweta yadav>cd Documents\jdk1.8.0_111\ShwetaDspract
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>javac ClientPrimeNum.java
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>java ClientPrimeNum
Enter a number:
04
04 is not a Prime Number
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>

C:\Windows\system32>cd..
C:\Windows>cd..
C:\>cd Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>javac ServerPrimeNum.java
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>java ServerPrimeNum
Server Started on port 8001
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>

```

## Example1(B)

**AIM:Write a program for implementing Client Server TCP based chatting application.**

### ClientChat.java

```

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

public class ClientChat {

```

```

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());
    }
}
}

```

### **ServerChat.java**

```

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

public class ServerChat {
    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;

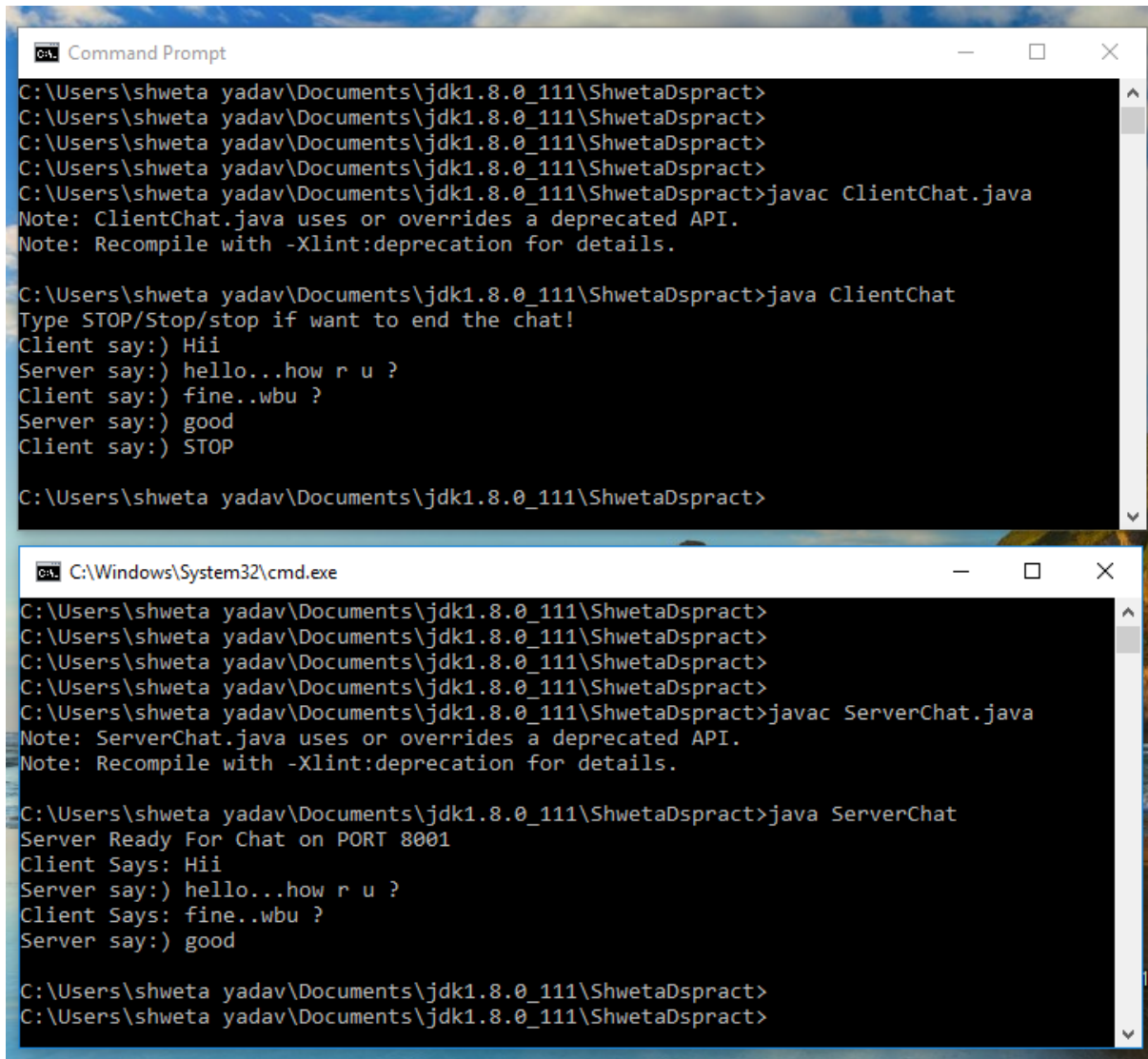
    while ((receive = inputStream.readLine()) != null) {
        if (receive.toLowerCase().equals("stop"))
            break;
        System.out.println("Client Says: " + receive);
        System.out.print("Server say:) ");
        send = inputBReader.readLine();
        outputStream.writeBytes(send + "\n");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

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

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

```

**Output :**



```
Command Prompt
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>javac ClientChat.java
Note: ClientChat.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>java ClientChat
Type STOP/Stop/stop if want to end the chat!
Client say:) Hii
Server say:) hello...how r u ?
Client say:) fine..wbu ?
Server say:) good
Client say:) STOP

C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>

C:\Windows\System32\cmd.exe
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>javac ServerChat.java
Note: ServerChat.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>java ServerChat
Server Ready For Chat on PORT 8001
Client Says: Hii
Server say:) hello...how r u ?
Client Says: fine..wbu ?
Server say:) good

C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>
```

## **PRACTICAL-2**

**AIM:** Write a program for implementing various arithmetic operations through Client and Server.

### **RMI (Remote Method Invocation)**

The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application 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*.

RMI uses stub and skeleton object for communication with the remote object.

### **stub**

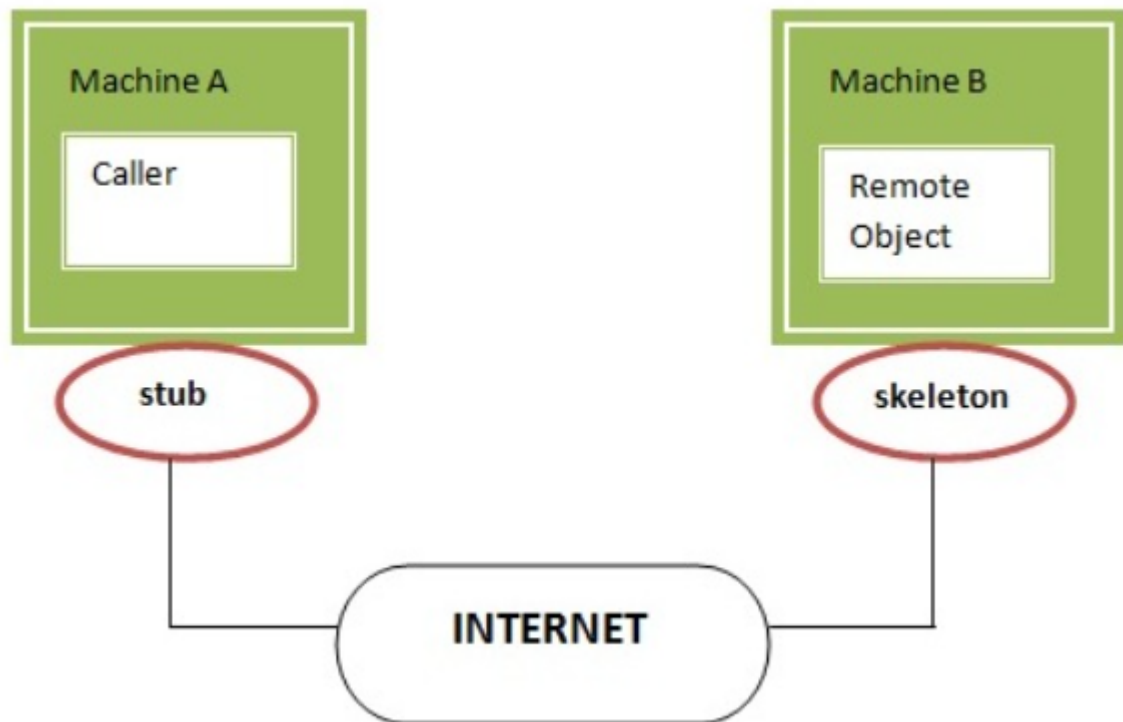
The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

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.



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

#### **RMInterfaceDate.java**

```
import java.rmi.*;
public interface RMInterfaceDate extends Remote
{
    public String printDate()throws Exception;
}
```

#### **RMIserverDate.java**

```
import java.rmi.*;
import java.rmi.server.*;
import java.util.Date;
import java.text.SimpleDateFormat;
public class RMIserverDate extends UnicastRemoteObject implements
RMInterfaceDate
{
    public RMIserverDate()throws Exception
    {
        System.out.println("Server is initiated");
    }
    public String printDate()throws Exception
```

```

{
    Date d = new Date();
    SimpleDateFormat myFormat = new SimpleDateFormat("dd-MM-yyyy
hh:mm:ss");
    String myDate = myFormat.format(d);
    System.out.println("Server:"+myDate);
    return myDate;
}
public static void main(String[] args) throws Exception
{
    System.out.println("RMIServerDate Started...");
    RMIServerDate obj = new RMIServerDate();
    Naming.bind("RMIServerDate",obj);
    System.out.println("Object registered..");
}
}

```

### **RMIClientDate.java**

```

import java.rmi.*;
class RMIClientDate
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("RMIClientDate Started..");
        RMIInterfaceDate
server=(RMIInterfaceDate)Naming.lookup("RMIServerDate");
        String serverDate = server.printDate();
        System.out.println("Server:"+serverDate);
    }
}

```

## Output :

CA: Command Prompt - rmiregistry

```
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\shweta yadav>cd..

C:\Users>cd..

C:\>cd ShwetaDsPract

C:\ShwetaDsPract>javac RMIInterfaceDate.java

C:\ShwetaDsPract>javac RMIClientDate.java

C:\ShwetaDsPract>javac RMIServerDate.java

C:\ShwetaDsPract>rmic RMIServerDate
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.
```

CA: Command Prompt - java RMIServerConvert

```
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\shweta yadav>cd..

C:\Users>cd..

C:\>cd ShwetaDsPract

C:\ShwetaDsPract>javac RMIServerDate.java

C:\ShwetaDsPract>java RMIServerDate
RMIServerDate Started...
Server is initiated
Object registered..
Server:26-02-2021 12:24:03
```

CA: Command Prompt

```
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\shweta yadav>cd..

C:\Users>cd..

C:\>cd ShwetaDsPract

C:\ShwetaDsPract>javac RMIClientDate.java

C:\ShwetaDsPract>java RMIClientDate
RMIClientDate Started..
Server:26-02-2021 12:24:03
```

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

### **RMIInterfaceConvert.java**

```
import java.rmi.*;
public interface RMIInterfaceConvert extends Remote
{
    public String ConvertToString(String number)throws Exception;
}
```

### **RMIserverConvert.java**

```
import java.rmi.*;
import java.rmi.server.*;
class RMIserverConvert extends UnicastRemoteObject implements
RMIInterfaceConvert
{
    public RMIserverConvert()throws Exception
    {
        System.out.println("RMIserverConvert is initialised");
    }
    public String ConvertToString(String number)throws Exception
    {
        String response = "";
        for(int i=0;i<number.length();i++)
        {
            int ascii=number.charAt(i);
            switch(ascii)
            {
                case 48:
                    response+="Zero";
                    break;
                case 49:
                    response+="One";
                    break;
                case 50:
                    response+="Two";
                    break;
                case 51:
                    response+="Three";
            }
        }
    }
}
```

```

        break;
        case 52:
            response+="Four";
            break;
        case 53:
            response+="Five";
            break;
        case 54:
            response+="Six";
            break;
        case 55:
            response+="Seven";
            break;
        case 56:
            response+="Eight";
            break;
        case 57:
            response+="Nine";
            break;
    }
}
return response;
}
public static void main(String[]args)throws Exception
{

    RMIServerConvert server = new RMIServerConvert();
    Naming.bind("RMIServerConvert",server);
    System.out.println("RMIServerConvert Object registered..");
}
}

```

### **RMIClientConvert.java**

```

import java.rmi.*;
import java.io.*;
class RMIClientConvert
{
    public static void main(String[] args)throws Exception

```



```

        {
            System.out.println("RMIClientConvert Started..");
            RMIInterfaceConvert
obj=(RMIInterfaceConvert)Naming.lookup("RMIServerConvert");
            BufferedReader br= new BufferedReader(new
InputStreamReader(System.in));
            System.out.println("Enter number:");
            String number = br.readLine();
            String response = obj.ConvertToString(number);
            System.out.println("The Word Representation of "+number+"digit
is:"+response);
        }
    }
}

```

## Output:

```

C:\ShwetaDsPract>rmiregistry

C:\ShwetaDsPract>javac RMIInterfaceConvert.java

C:\ShwetaDsPract>javac RMIClientConvert.java

C:\ShwetaDsPract>javac RMIServerConvert.java

C:\ShwetaDsPract>rmic RMIServerConvert
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

C:\ShwetaDsPract>rmiregistry

```

```

C:\ShwetaDsPract>javac RMIServerConvert.java

C:\ShwetaDsPract>java RMIServerConvert
RMIServerConvert is initialised
RMIServerConvert Object registered..

```

```
C:\ShwetaDsPract>javac RMIClientConvert.java

C:\ShwetaDsPract>java RMIClientConvert
RMIClientConvert Started..
Enter number:
5
The Word Representation of 5digit is:Five

C:\ShwetaDsPract>
```

### **PRACTICAL- 3**

**Aim: Show the implementation of Remote Procedure Call.**

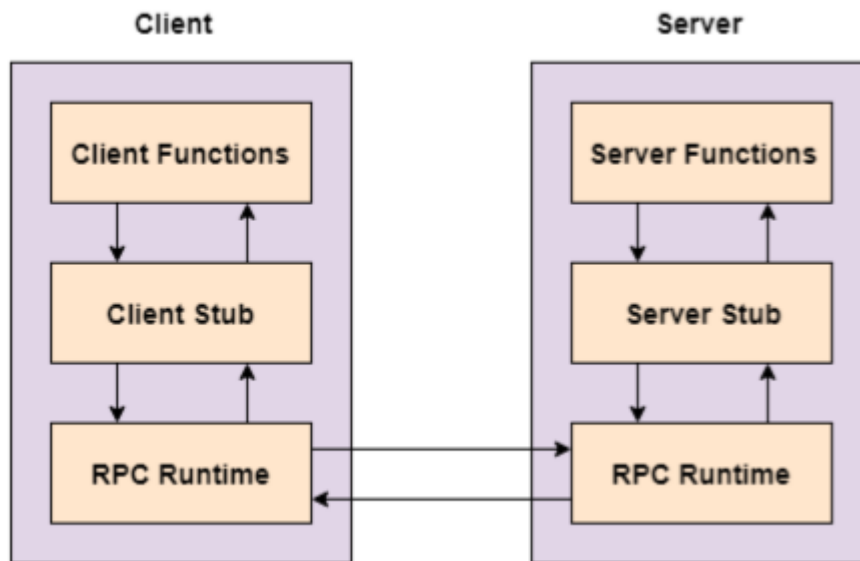
**Remote Procedure Call (RPC):**

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

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

### RPCServer.java

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

```
final class RPCServer
```

```

{
    DatagramSocket ds;
    DatagramPacket dp;
    String str,methodName,result;
    int val1,val2;
    RPCServer()
    {
        try
        {
            ds=new DatagramSocket(1200);
            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);
                }
            }
            else
            {

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

                int i=0;

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

            InetAddress ia = InetAddress.getLocalHost();
            if(methodName.equalsIgnoreCase("add"))
            {

```

```

        result= "" + add(val1,val2);
    }
    else if(methodName.equalsIgnoreCase("sub"))
    {
        result= "" + sub(val1,val2);
    }
    else if(methodName.equalsIgnoreCase("mul"))
    {
        result= "" + mul(val1,val2);
    }
    else if(methodName.equalsIgnoreCase("div"))
    {
        result= "" + div(val1,val2);
    }
    byte b1[]=result.getBytes();

    DatagramSocket ds1 = new DatagramSocket();

    DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);

    System.out.println("result : "+result+"\n"); ds1.send(dp1);
    }
}
catch (Exception e)
{
    e.printStackTrace();
}
}
public int add(int val1, int val2)
{
    return val1+val2;
}
public int sub(int val3, int val4)
{
    return val3-val4;
}
public int mul(int val3, int val4)
{
    return val3*val4;
}

```

```

    }
    public int div(int val3, int val4)
    {
        return val3/val4;
    }

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

```

## **RPCClient.java**

```

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

class RPCClient
{
    RPCClient()
    {
        try
        {
            InetAddress ia = InetAddress.getLocalHost();
            DatagramSocket ds = new DatagramSocket();
            DatagramSocket ds1 = new DatagramSocket(1300);
            System.out.println("\nRPC Client\n");
            System.out.println("Enter method name and parameter like add num1
num2\n");
            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,ia,1200);
                ds.send(dp);
                dp = new DatagramPacket(b,b.length);
                ds1.receive(dp);
                String s = new String(dp.getData(),0,dp.getLength());
                System.out.println("\nResult = " + s + "\n");
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}

```

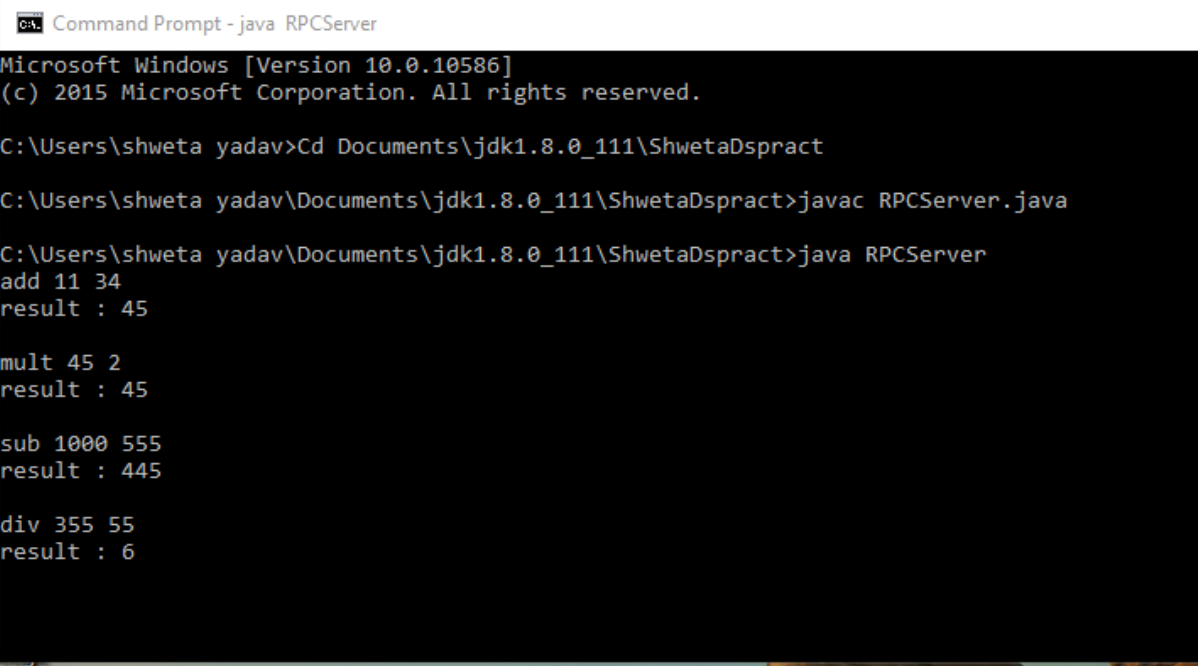
```

        }
        //ds.close();
        //ds1.close();

    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}
public static void main(String[] args)
{
    new RPCCClient();
}
}

```

## Output:



The screenshot shows a Windows Command Prompt window titled "Command Prompt - java RPCServer". The window displays the following text:

```

Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\shweta yadav>Cd Documents\jdk1.8.0_111\ShwetaDspract

C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>javac RPCServer.java

C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>java RPCServer

add 11 34
result : 45

mult 45 2
result : 45

sub 1000 555
result : 445

div 355 55
result : 6

```

```
Command Prompt - java RPCClient

Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\shweta yadav>Cd Documents\jdk1.8.0_111\ShwetaDspract

C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>javac RPCClient.java

C:\Users\shweta yadav\Documents\jdk1.8.0_111\ShwetaDspract>java RPCClient

RPC Client

Enter method name and parameter like add num1 num2

add 11 34

Result = 45

mult 45 2

Result = 45

sub 1000 555

Result = 445

div 355 55

Result = 6
```

### Example 3B

**AIM:** A program that finds the square, square root, cube and cube root of the entered number.

#### RPCServersqrt.java

```
import java.io.*;
import java.net.*;
import java.util.StringTokenizer;
import java.util.*;
final class RPCServerSqrt{
    DatagramSocket ds;
    DatagramPacket dp;
    String str,methodName,result;
    int val1,val2;
    RPCServerSq()
```



```

{
    try
    {
        ds=new DatagramSocket(1200);
        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);
            }
            else
            {
                StringTokenizer st = new StringTokenizer(str," ");
                int i=0;
                while(st.hasMoreTokens())
                {
                    String token=st.nextToken();
                    methodName=token;
                    val1 = Integer.parseInt(st.nextToken());
                }
            }
            System.out.println(str);

            InetAddress ia = InetAddress.getLocalHost();
            if(methodName.equalsIgnoreCase("sqr"))
            {
                result= "" + sqr(val1);
            }
            else if(methodName.equalsIgnoreCase("sqrroot"))
            {
                result= "" + sqrroot(val1);
            }
        }

        else if(methodName.equalsIgnoreCase("cube"))
        {
            result= "" + cube(val1);
        }
    }
}

```

```

        else if(methodName.equalsIgnoreCase("cuberoot"))
        {
            result= "" + cuberoot(val1);
        }
        byte b1[]=result.getBytes();

        DatagramSocket ds1 = new DatagramSocket();
        DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);

        System.out.println("result : "+result+"\n"); ds1.send(dp1);
    }
}
catch (Exception e)
{
    e.printStackTrace();
}
}
public int sqr(int val1)
{
    return val1*val1;
}
public double sqrroot(int val3)
{
    double n=Math.sqrt(val3);
    return n;
}
public int cube(int val1)
{
    return val1*val1*val1;
}
    public double cuberoot(int val1)
    {
        double c=Math.cbrt(val1);
        return c;
    }

public static void main(String[] args)
{
    new RPCServerSqrt();

```

```
}  
  
}
```

## **RPCClientSqrt.java**

```
import java.io.*;  
import java.net.*;  
class RPCClientSqrt  
{  
    RPCClientSqrt()  
    {  
        try  
        {  
            InetAddress ia = InetAddress.getLocalHost();  
            DatagramSocket ds = new DatagramSocket();  
            DatagramSocket ds1 = new DatagramSocket(1300);  
            System.out.println("\nRPC Client\n");  
            System.out.println("Enter method name and parameter like cube num1 \n");  
            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,ia,1200);  
                ds.send(dp);  
                dp = new DatagramPacket(b,b.length);  
                ds1.receive(dp);  
                String s = new String(dp.getData(),0,dp.getLength());  
                System.out.println("\nResult = " + s + "\n");  
            }  
  
            //ds.close();  
            //ds1.close();  
  
        }  
        catch (IOException e)  
        {  
            e.printStackTrace();  
        }  
    }  
}
```

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

**Output :**

Command Prompt - java RPCServerSqrt

```
Microsoft Windows [Version 10.0.18363.1379]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\shweta yadav>cd..

C:\Users>cd..

C:\>cd ShwetaDsPract

C:\ShwetaDsPract>javac RPCServerSqrt.java

C:\ShwetaDsPract>java RPCServerSqrt
java.util.NoSuchElementException
    at java.util.StringTokenizer.nextToken(Unknown Source)
    at RPCServerSqrt.<init>(RPCServerSqrt.java:33)
    at RPCServerSqrt.main(RPCServerSqrt.java:89)

C:\ShwetaDsPract>java RPCServerSqrt
sqr 2
result : 4

cube 9
result : 729

cuberoot 8
result : 2.0
```

Command Prompt - java RPCClientSqrt

```
C:\ShwetaDsPract>
C:\ShwetaDsPract>
C:\ShwetaDsPract>java RPCClientSqrt

RPC Client

Enter method name and parameter like cube num1

sqr 2

Result = 4

cube 9

Result = 729

cuberoot 8

Result = 2.0
```

## **PRACTICAL-4**

**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).

23

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

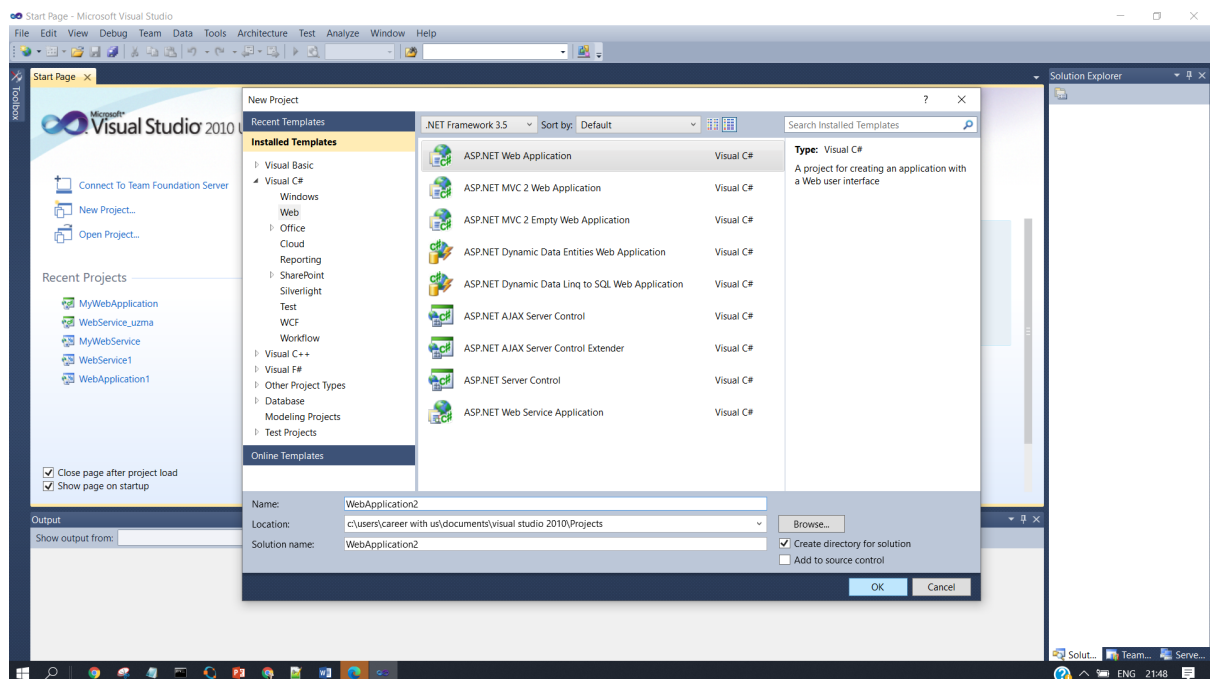
A web service can perform almost any kind of task

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.

## Steps to Create Web services:

### Create Web Service

Step 1: Create a ASP.Net Web Application by clicking File, NewProject. And select ASP.NET Web Application.



Step 2: Now type the below code in Default.aspx.cs file as shown in below screenshot.



```
Default.aspx.cs X Default.aspx
WebApplication2_Default

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication2
{
    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);
            Label4.Text = c.ToString();
        }
    }
}
```

```
Default.aspx.cs X Default.aspx
WebApplication2_Default

    }

    protected void Button2_Click(object sender, EventArgs e)
    {
        a = Convert.ToInt32(TextBox1.Text);
        b = Convert.ToInt32(TextBox2.Text);
        c = obj.sub(a, b);
        Label4.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);
        Label4.Text = c.ToString();
    }

    protected void Button4_Click(object sender, EventArgs e)
    {
        a = Convert.ToInt32(TextBox1.Text);
        b = Convert.ToInt32(TextBox2.Text);
        c = obj.div(a, b);
        Label4.Text = c.ToString();
    }
}
```

**Code:**

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using WebApplication1.localhost;

namespace WebApplication1
{
    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);
        }
    }
}
```

```

        Label1.Text = c.ToString();
    }

protected void BUTTONSUBTRACT_Click(object sender, EventArgs e)
{
    a = Convert.ToInt32(TextBox1.Text);
    b = Convert.ToInt32(TextBox2.Text);
    c = obj.sub(a,b);
    Label1.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);
    Label1.Text = c.ToString();
}

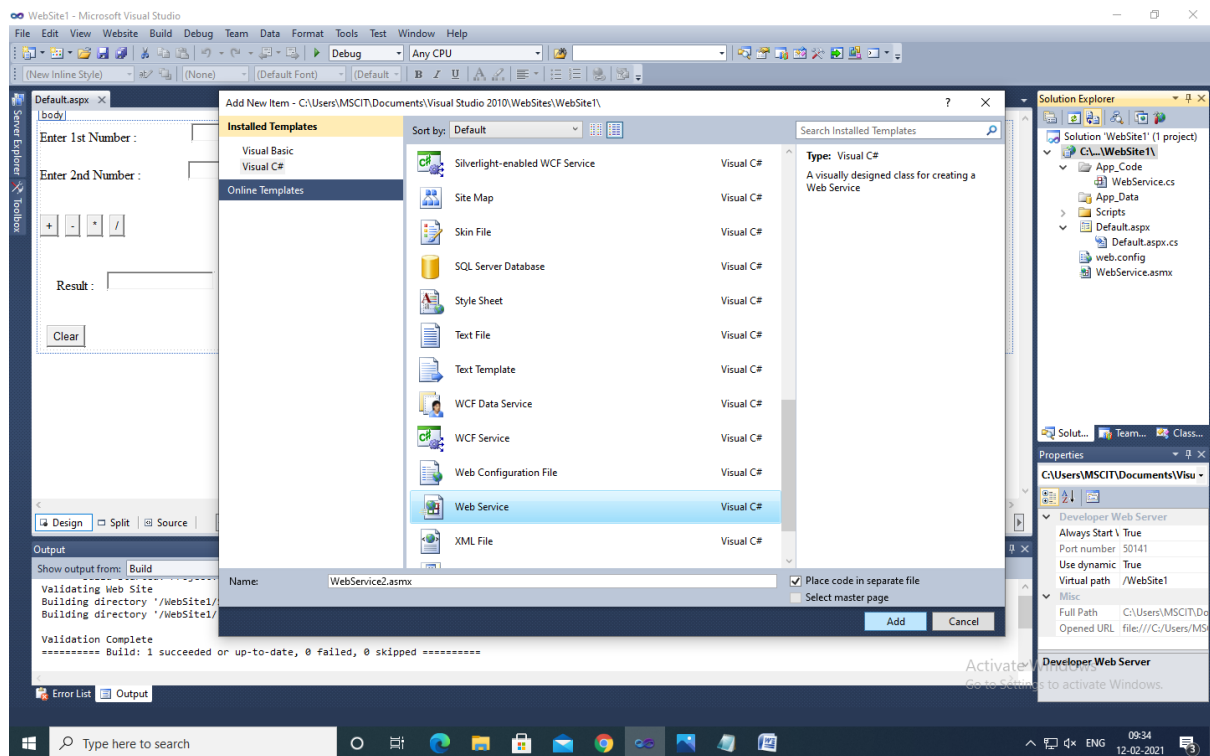
protected void Buttondivide_Click(object sender, EventArgs e)
{
    a = Convert.ToInt32(TextBox1.Text);
    b = Convert.ToInt32(TextBox2.Text);
    c = obj.div(a,b);
    Label1.Text = c.ToString();
}
}

```

}

Step 3: now create a aspx file

Step 4: Now Design a simple calculator in default.aspx file in source section as shown below.



Step 4: Now Design a simple calculator by writing code .

Step 5: save and Debug the project. Output will be shown in browser .

**Output:**

< > ↺ ☐

🌐

localhost:50941/WebForm1.aspx

Number 1	<input type="text" value="25"/>	<input type="radio"/> Add
Number 2	<input type="text" value="16"/>	<input type="radio"/> Sub
	<input type="button" value="Calculate"/>	<input type="radio"/> Mul
		<input checked="" type="radio"/> Div
Result	1,5625	

## **PRACTICAL-5**

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

### **Mutual exclusion**

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.

### **Example 5A**

**AIM: Write a program to execute any one mutual exclusion algorithm**

## TokenServer.java

```
import java.net.*;

public class TokenServer
{
    public static void main(String args[])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);

}

}

```

### **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;

}

}

```

```
}
```

## **TokenClient2.java**

```
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.setRecPort(8004);

            tkcl.setSendPort(9002);

            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;

}

}

}
```

**Output :**

CA Command Prompt - java TokenServer

Microsoft Windows [Version 10.0.18363.1379]  
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\shweta yadav>cd..

C:\Users>cd..

C:\>cd ShwetaDsPract

C:\ShwetaDsPract>javac TokenServer.java

C:\ShwetaDsPract>java TokenServer

The message is ClientOne.....hi ...how are you doing ??

The message is ClientOne.....hii....whats up ??

The message is ClientTwo.....i am client2

The message is ClientOne.....i am client 1

The message is ClientTwo.....i quit

Command Prompt - java TokenClient1

```
C:\ShwetaDsPract>javac TokenClient1.java
```

```
C:\ShwetaDsPract>java TokenClient1
```

```
Do you want to enter the Data -->YES/NO
```

```
yes
```

```
ready to send
```

```
Enter the Data
```

```
hi ...how are you doing ??
```

```
now sending.....
```

```
Data Sent
```

```
Do you want to enter the Data -->YES/NO
```

```
yes
```

```
ready to send
```

```
Enter the Data
```

```
hii....whats up ??
```

```
now sending.....
```

```
Data Sent
```

```
Do you want to enter the Data -->YES/NO
```

```
no
```

```
The data is Token
```

```
Do you want to enter the Data -->YES/NO
```

```
yes
```

```
ready to send
```

```
Enter the Data
```

```
i am client 1
```

```
now sending.....
```

```
Data Sent
```

```
Do you want to enter the Data -->YES/NO
```

```
no
```

Command Prompt - java TokenClient2

```
entering recieving mode
The data is Token
Do you want to enter the Data -->YES/NO

Do you want to enter the Data -->YES/NO
yes
ready to send
Enter the Data
i am client2
now sending.....
Data Sent
Do you want to enter the Data -->YES/NO
yes
Do you want to enter the Data -->YES/NO
i quit
Do you want to enter the Data -->YES/NO
no
The data is Token
Do you want to enter the Data -->YES/NO
yes
ready to send
Enter the Data
i quit
now sending.....
Data Sent
Do you want to enter the Data -->YES/NO
```

## **PRACTICAL-6**

**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.

### **BulyAlgo.java**

```
import java.io.*;
import java.util.Scanner;

class BulyAlgo{
    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,j,k,l,m;
```

```
    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 initiate 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

```
8  
Priority  
2  
Which process will initiate election?  
3  
Election message is sent from 3 to 5  
Final coordinator is 5  
  
C:\ShwetaDsPract>javac BulyAlgo.java  
  
C:\ShwetaDsPract>java BulyAlgo  
Enter the number of process  
4  
For process 1:  
Status:  
10  
Priority  
3  
For process 2:  
Status:  
6  
Priority  
5  
For process 3:  
Status:  
9  
Priority  
5  
For process 4:  
Status:  
4  
Priority  
6  
Which process will initiate election?  
4  
Final coordinator is 4  
  
C:\ShwetaDsPract>
```

## **PRACTICAL-7**

**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.

### **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) {
        try
        {
            ServerSocket ss = new ServerSocket(8001);
            System.out.println("Server is accept message....");
            Socket s = ss.accept();
            DataInputStream in = new DataInputStream(s.getInputStream());
            String receieve;
            DateFormat dateFormat = new SimpleDateFormat("hh:mm:ss.SSSS");
            while((receieve = in.readLine()) != null)
            {
                String[] mesaage = receieve.split(",");
```



```

if(mesaage[0].equals("stop") || mesaage[0].equals("STOP") ||
mesaage[0].equals("Stop"))
break;
Date date = new Date();
Long clienttime = Long.parseLong(mesaage[1]);
Long timeMilli = date.getTime();
Long requiretime =timeMilli-clienttime;
System.out.println("This is the message: "+mesaage[0]);
System.out.println("Server will not accept the message if its taken more than 2
milisecond.");
if(clienttime.equals(timeMilli))
{
String strDate = dateFormat.format(timeMilli);
System.out.println("Message sending time and receving time is same:"+strDate);
}
else if(requiretime > 2)
{
String clinttim = dateFormat.format(clienttime);
System.out.println("Messange sending time from client:"+ clinttim+"\n");
String strDate = dateFormat.format(timeMilli);
System.out.println("Messange received from client to:"+strDate+"\n");
System.out.println("This message is rejected.");
}
else
{
String clinttim = dateFormat.format(clienttime);
System.out.println("Messange sending time from client:"+ clinttim+"\n");
String strDate = dateFormat.format(timeMilli);
System.out.println("Messange received from client to:"+strDate+"\n");
System.out.println("This message is accepted.");
}
}
in.close();
}
catch(Exception e)
{
System.out.println(e.toString());
}
}
}

```

## **SCClient.java**


```
import java.net.*;
import java.io.*;
import java.util.Date;
public class SCClient {
public static void main(String[] args) {
try
{
Socket cs = new Socket("LocalHost",8001);
BufferedReader infu = new BufferedReader(new InputStreamReader(System.in));
DataOutputStream ot = new DataOutputStream(cs.getOutputStream());
String send;
System.out.println("Type a message to send into sever");
while((send = infu.readLine()) != null )
{
Date date = new Date();
long timeMilli = date.getTime();
String t=String.valueOf(timeMilli);
ot.writeBytes(send+" "+t+"\n");
if(send.equals("stop") || send.equals("STOP") || send.equals("Stop"))
break;
}
infu.close();
ot.close();
cs.close();
}
catch(Exception e)
{
System.out.println(e.toString());
}
}
}
```

## **Output:**

```
C:\ShwetaDsPract>java SCServer
Server is accept message....
This is the message: hii
Server will not accept the message if its taken more than 2 milisecond.
Message sending time and receving time is same:02:48:26.0823
This is the message: ddd
Server will not accept the message if its taken more than 2 milisecond.
Message sending time and receving time is same:02:48:53.0004
This is the message: how r u ??
Server will not accept the message if its taken more than 2 milisecond.
Messange sending time from client:02:49:05.0491

Messange received from client to:02:49:05.0507

This message is rejected.
```

 Command Prompt - java SCClient

```
hi server

C:\ShwetaDsPract>java SCClient
java.net.ConnectException: Connection refused: connect

C:\ShwetaDsPract>java SCClient
Type a message to send into sever
hii
ddd
how r u ??
```

## **PRACTICAL-8**

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

### **TPCServer.java**

```
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);

}

}

```

### **TPCSClient1.java**

```

import java.io.*;

import java.net.*;

public class TPCClient1

{

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

    {

        InetAddress lclhost;

        lclhost=InetAddress.getLocalHost();

        Client cInt=new Client(lclhost);

        cInt.setSendPort(9001);

        cInt.setRecPort(8000);

        cInt.recData();

        cInt.sendData();
    }
}

```



```

        cInt.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);

```

```

    }
}

TPCSCClient2.java

import java.io.*;

import java.net.*;

public class TPCClient2{

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

        InetAddress lclhost;

        lclhost = InetAddress.getLocalHost();

        Client client = new Client(lclhost);

        //Sending data to client2

        client.setSendPort(9003); //recport = 8002

        client.setRecPort(8002); //senport = 9003

        client.recData();

        client.sendData();

        client.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(msgStr);

    }

};

```

**Output :**

Command Prompt

```
C:\ShwetaDsPract>
C:\ShwetaDsPract>javac TPCServer.java
C:\ShwetaDsPract>java TPCServer
Server in sending mode....
Send request data to client1....
Data is null
C:\ShwetaDsPract>java TPCServer
Server in sending mode....
Send request data to client1....
hii
Data is hii
Waiting for response from client1.....
String=hiedfhjivdfhidr
Counter value =0n Send value = 1
Send request data to client2...
shwetaa
Data is shwetaa
Waiting for response from client2...
String=VOTE_COMMIT
Counter value =1n Send value = 2
C:\ShwetaDsPract>h
```

CA Command Prompt

Microsoft Windows [Version 10.0.18363.1379]  
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\shweta yadav>cd..

C:\Users>cd..

C:\>cd ShwetaDsPract

C:\ShwetaDsPract>javac TPCCClient1.java

C:\ShwetaDsPract>java TPCCClient1

C:\ShwetaDsPract>java TPCCClient1

Client1 dataVOTE\_REQUEST

Enter the Response 'VOTE\_COMMIT' || 'VOTE\_ABORT'

hiedfhjivdfhldr

Data is : hiedfhjivdfhldr

Client1 dataGLOBAL\_ABORTTRANSACTION ABORTED

C:\ShwetaDsPract>h

CA Command Prompt

```
Microsoft Windows [Version 10.0.18363.1379]  
(c) 2019 Microsoft Corporation. All rights reserved.
```

```
C:\Users\shweta yadav>cd..
```

```
C:\Users>cd..
```

```
C:\>cd ShwetaDsPract
```

```
C:\ShwetaDsPract>javac TPCClient2.java
```

```
C:\ShwetaDsPract>java TPCClient2
```

```
C:\ShwetaDsPract>java TPCClient2
```

```
Client1 dataVOTE_REQUEST
```

```
Enter the Response 'VOTE_COMMIT' || 'VOTE_ABORT'
```

```
VOTE_COMMIT
```

```
Data is : VOTE_COMMIT
```

```
Client1 dataGLOBAL_ABORTTRANSACTION ABORTED
```

```
C:\ShwetaDsPract>
```

```
C:\ShwetaDsPract>
```