



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

**DEPARTMENT OF INFORMATION  
TECHNOLOGY  
2020 - 2021**

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

**Name : Neha Dattatray Dawale  
Roll No.: 06**



Hindi Vidya Prachar Samiti's

**RAMNIRANJAN**

**JHUNJHUNWALA COLLEGE**  
**(AUTONOMOUS)**



Opposite Ghatkopar Railway Station, Ghatkopar West, Mumbai-400086

## CERTIFICATE

This is to certify that Miss. **Neha Dattatray Dawale** with Roll No.**06** 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>Practical 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 No: 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.**

### **Code:**

#### **TCPServerPrimeNumber .java**

```
import java.net.*;
import java.io.*;
public class TCPServerPrimeNumber {
    public static void main(String[] args){
        try {
            ServerSocket ss=new ServerSocket(8001);
            System.out.println("Server started");
            Socket s=ss.accept();
            DataInputStream in =new DataInputStream(s.getInputStream());
            int x=in.readInt();
            DataOutputStream otc=new DataOutputStream (s.getOutputStream());
            int y=x/2;
            if(x==1){
                otc.writeUTF(x+" is neither prime nor not");
                System.exit(0);
            }
        }
    }
}
```

```

    }
    if(x==2 || x==3){
        otc.writeUTF(x+" is a prime");
        System.exit(0);
    }
    for(int i=2;i<=y;i++){
        if(x%i !=0){
            otc.writeUTF(x+" is a prime");
        }
        else{
            otc.writeUTF(x+" is not prime");
        }
    }
}
} catch (Exception e){
    System.out.println(e.toString());
}
}
}

```

## **TCPClientPrimeNumber.java**

### **Code:**

```

import java.net.*;

import java.io.*;

public class TCPClientPrimeNumber
{
    public static void main(String args[])
    {
        try
        {
            Socket cs = new Socket("LocalHost", 8001);

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

            System.out.println("Enter a number: ");

            int x = Integer.parseInt(input.readLine());

            DataOutputStream out = new DataOutputStream(cs.getOutputStream());

            out.writeInt(x);

            DataInputStream in = new DataInputStream(cs.getInputStream());

```

```

        System.out.println(in.readUTF());

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

```

### Output:

The image displays two screenshots of a Windows command prompt window, titled "C:\Windows\System32\cmd.exe".

**Top Screenshot:**

```

Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Neha Dawale MscIT(Sem I)>set path="C:\Program Files\Java\jdk-13.0.1\bin"

C:\Neha Dawale MscIT(Sem I)>javac TCPServerPrimeNumber.java

C:\Neha Dawale MscIT(Sem I)>java TCPServerPrimeNumber
Server started

C:\Neha Dawale MscIT(Sem I)>java TCPServerPrimeNumber
Server started

C:\Neha Dawale MscIT(Sem I)>_

```

**Bottom Screenshot:**

```

Select C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>set path="C:\Program Files\Java\jdk-13.0.1\bin"

C:\Neha Dawale MscIT(Sem I)>javac TCPClientPrimeNumber.java

C:\Neha Dawale MscIT(Sem I)>java TCPClientPrimeNumber
Enter a number:
3
3 is a prime

C:\Neha Dawale MscIT(Sem I)>java TCPClientPrimeNumber
Enter a number:
6
6 is not prime

C:\Neha Dawale MscIT(Sem I)>

```

## **PRACTICAL 1(B)**

**AIM: A client server TCP based chatting application.**

**Code:**

**TCPServerChat.java**

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

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**

#### **Code:**

```

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:

```
C:\Windows\System32\cmd.exe
C:\Neha Dawale MscIT(Sem I)>javac TCPServerChat.java
Note: TCPServerChat.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Neha Dawale MscIT(Sem I)>java TCPServerChat
Server Ready For Chat on PORT 8001
Client Says: hii
Server say:) hello
Client Says: how r u
Server say:) great
Client Says: who is this
Server say:) this is neha dawale

C:\Neha Dawale MscIT(Sem I)>
```

```
C:\Windows\System32\cmd.exe
C:\Neha Dawale MscIT(Sem I)>javac TCPClientChat.java
Note: TCPClientChat.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Neha Dawale MscIT(Sem I)>java TCPClientChat
Type STOP/Stop/stop if want to end the chat !
Client say:) hii
Server say:) hello
Client say:) how r u
Server say:) great
Client say:) who is this
Server say:) this is neha dawale
Client say:) stop

C:\Neha Dawale MscIT(Sem I)>
```

## Practical No:02

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

**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. A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

### **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),

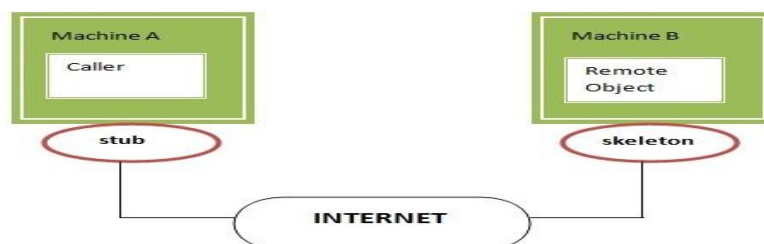
### **skeleton**

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

It reads the parameter for the remote method

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

In the Java 2 SDK, an stub protocol was introduced that eliminates the need for skeletons.



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

### **Code:**

#### **InterDate.java**

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

#### **ServerDate.java**

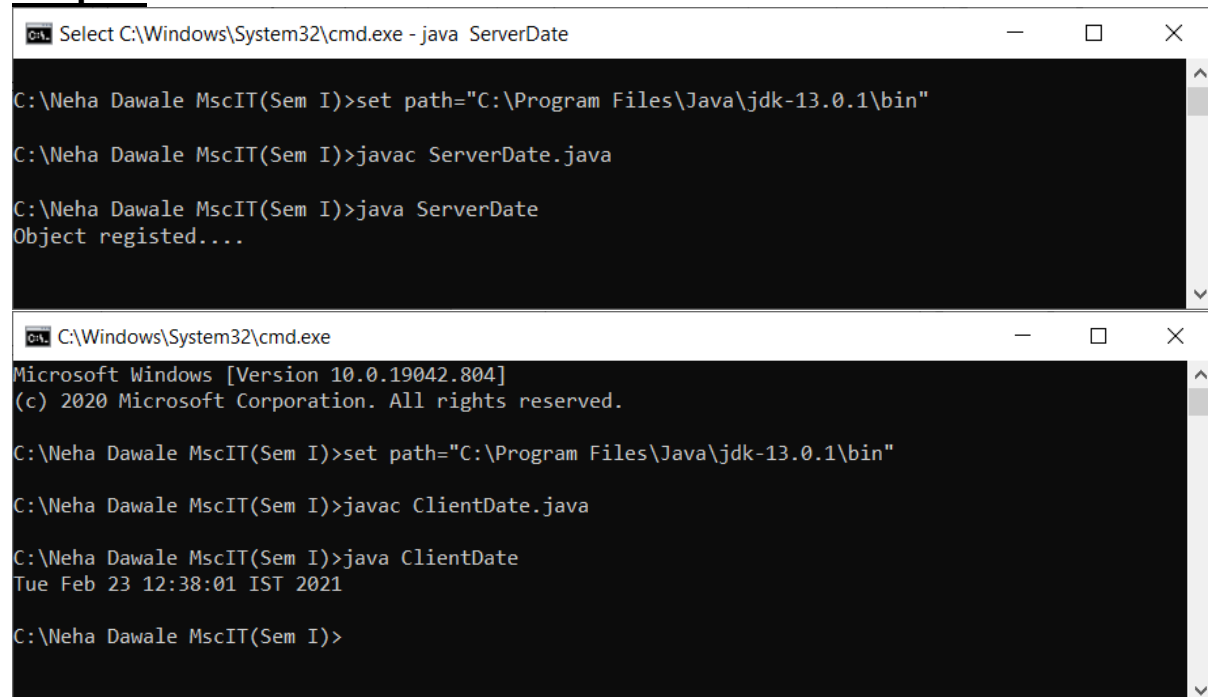
```
import java.rmi.*;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.rmi.server.*;
import java.util.*;
public class ServerDate extends UnicastRemoteObject implements InterDate{
    public ServerDate() throws RemoteException
    {
    }
    public String display() 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 registered....");
        }
        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);
        }
    }
}
```

## Output:



```
C:\Windows\System32\cmd.exe - java ServerDate
C:\Neha Dawale MscIT(Sem I)>set path="C:\Program Files\Java\jdk-13.0.1\bin"
C:\Neha Dawale MscIT(Sem I)>javac ServerDate.java
C:\Neha Dawale MscIT(Sem I)>java ServerDate
Object registred....

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.804]
(c) 2020 Microsoft Corporation. All rights reserved.
C:\Neha Dawale MscIT(Sem I)>set path="C:\Program Files\Java\jdk-13.0.1\bin"
C:\Neha Dawale MscIT(Sem I)>javac ClientDate.java
C:\Neha Dawale MscIT(Sem I)>java ClientDate
Tue Feb 23 12:38:01 IST 2021
C:\Neha Dawale MscIT(Sem I)>
```

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

### **InterConvert.java**

```
import java.rmi.*;

public interface InterConvert extends Remote{

    public String display(String n) throws RemoteException;

}
```

### **ServerConvert.java**

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

public class ServerConvert extends UnicastRemoteObject implements InterConvert{

    public ServerConvert() throws RemoteException
    {
    }

    public String display(String n) throws RemoteException
    {
        String str1 = "";
        for(int i=0;i<n.length();i++)

        {

            int ascii=n.charAt(i);

            switch(ascii)

            {

                case 48:

                    str1 += "ZERO";
```

```
        break;

        case 49:

            str1 += "ONE";

            break;

        case 50:

            str1 += "TWO";

            break;

        case 51:

            str1 += "THREE";

            break;

        case 52:

            str1 += "FOUR";

            break;

        case 53:

            str1 += "FIVE";

            break;

        case 54:

            str1 += "SIX";

            break;

        case 55:

            str1 += "SEVEN";

            break;

        case 56:

            str1 += "EIGHT";

            break;

        case 57:

            str1 += "NINE";

            break;

    }
```

```

    }
    return str1;
    }

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

```

### **ClientConvert.java**

```

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

public class ClientConvert{

    public static void main(String args[]) throws RemoteException{

        ClientConvert c1 = new ClientConvert();

        c1.connectRemote();
    }
}

```



```
    }  
    private void connectRemote() throws RemoteException{  
        try{  
            BufferedReader info=new BufferedReader(new InputStreamReader(System.in));  
            System.out.println("Enter Number");  
            String s1=info.readLine();  
            String s2;  
            Registry reg = LocateRegistry.getRegistry("localhost",5678);  
            InterConvert h1 = (InterConvert)reg.lookup("DS");  
            s2 = h1.display(s1);  
            System.out.println(s2);  
        }  
        catch(Exception e){  
            System.out.println(e);  
        }  
    }  
}
```

## Output:

```
C:\Windows\System32\cmd.exe - java ServerConvert
Microsoft Windows [Version 10.0.19042.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Neha Dawale MscIT(Sem I)>set path="C:\Program Files\Java\jdk-13.0.1\bin"

C:\Neha Dawale MscIT(Sem I)>javac ServerConvert.java

C:\Neha Dawale MscIT(Sem I)>java ServerConvert
Object registered....
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.804]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Neha Dawale MscIT(Sem I)>set path="C:\Program Files\Java\jdk-13.0.1\bin"

C:\Neha Dawale MscIT(Sem I)>javac ClientConvert.java

C:\Neha Dawale MscIT(Sem I)>java ClientConvert
Enter Number
123
ONETWOTHREE

C:\Neha Dawale MscIT(Sem I)>java ClientConvert
Enter Number
578
FIVESEVENEIGHT

C:\Neha Dawale MscIT(Sem I)>_
```

## Practical No: 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.

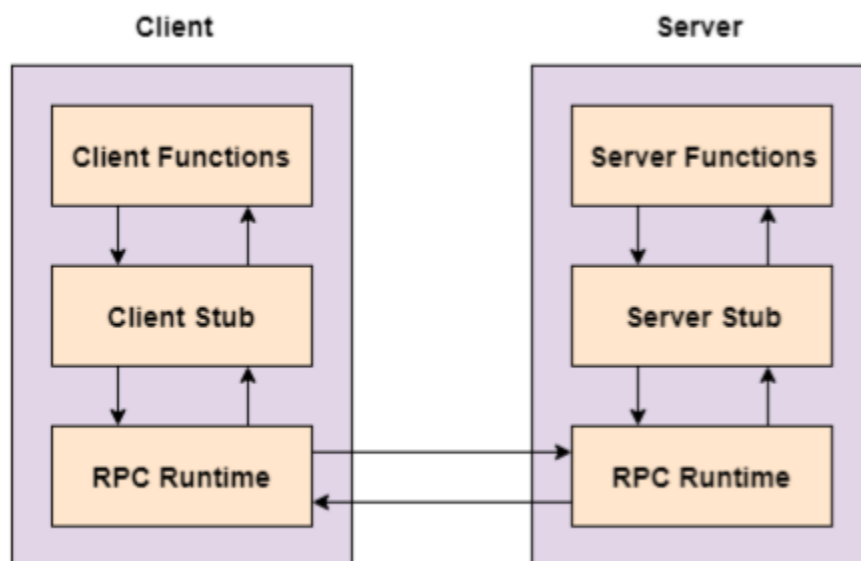
A client has a request message that the RPC translates and sends to the server.

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.

**Code:**

**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");

    }


    //ds.close();

    //ds1.close();

}

catch (IOException e)

{

    e.printStackTrace();

}

}

public static void main(String [] args)

{

    new RPCCClient();

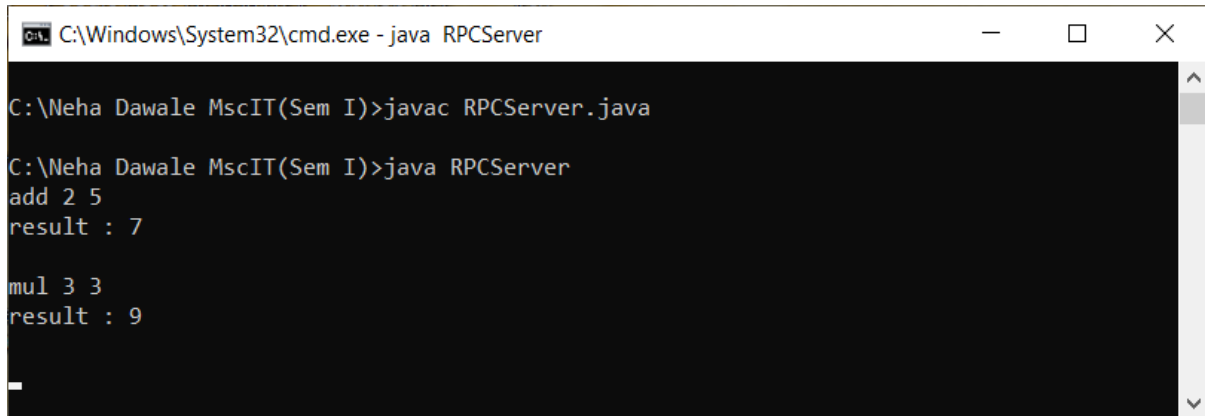
}

}

```



## Output:

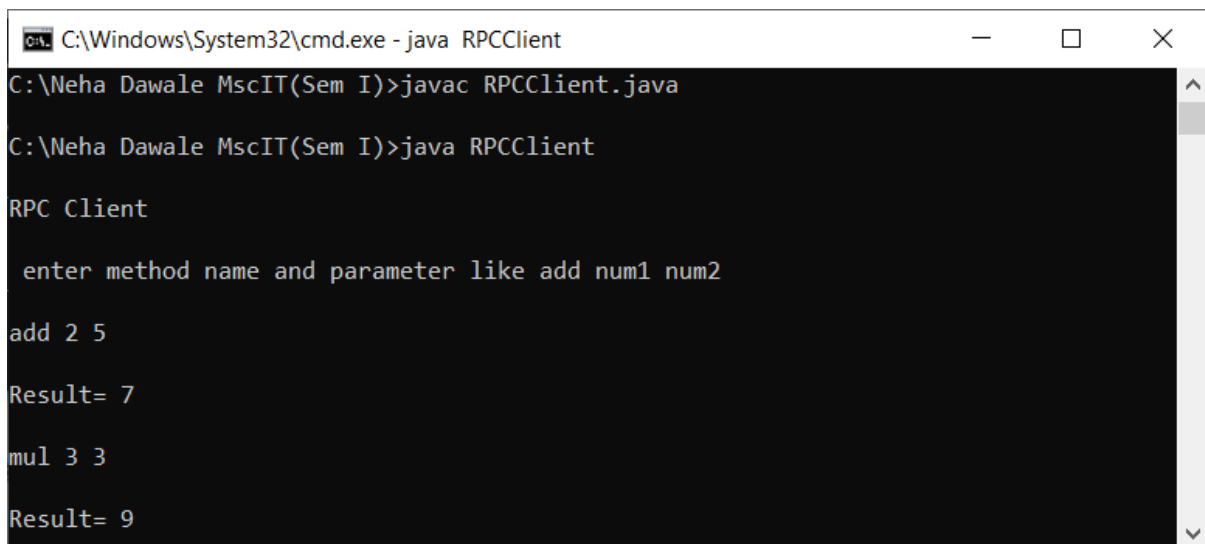


```
C:\Windows\System32\cmd.exe - java RPCServer

C:\Neha Dawale MscIT(Sem I)>javac RPCServer.java

C:\Neha Dawale MscIT(Sem I)>java RPCServer
add 2 5
result : 7

mul 3 3
result : 9
```



```
C:\Windows\System32\cmd.exe - java RPCClient

C:\Neha Dawale MscIT(Sem I)>javac RPCClient.java

C:\Neha Dawale MscIT(Sem I)>java RPCClient

RPC Client

enter method name and parameter like add num1 num2

add 2 5

Result= 7

mul 3 3

Result= 9
```

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

**Code:**

**RPCServerSq.java**

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

final class RPCServerSq
{
    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 RPCServerSq();
    }
}

```

### **RPCClientSq.java**

```

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

class RPCClientSq
{
    RPCClientSq()
    {
        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 RPCCClientSq();

}

}

```

## Output:

```
C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>javac RPCServerSq.java

C:\Neha Dawale MscIT(Sem I)>java RPCServerSq
sqr 5
result : 25

sqrrrot 3
result : 25

cube 3
result : 27

cuberoot 6
result : 1.8171205928321397

C:\Neha Dawale MscIT(Sem I)>_
```

```
C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>javac RPCClientSq.java

C:\Neha Dawale MscIT(Sem I)>java RPCClientSq

RPC Client

  enter method name and parameter like cube num1

sqr 5
Result= 25

sqrrrot 3
Result= 25

cube 3
Result= 27

cuberoot 6
Result= 1.81712059

q

C:\Neha Dawale MscIT(Sem I)>_
```

## **Practical No: 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).

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.

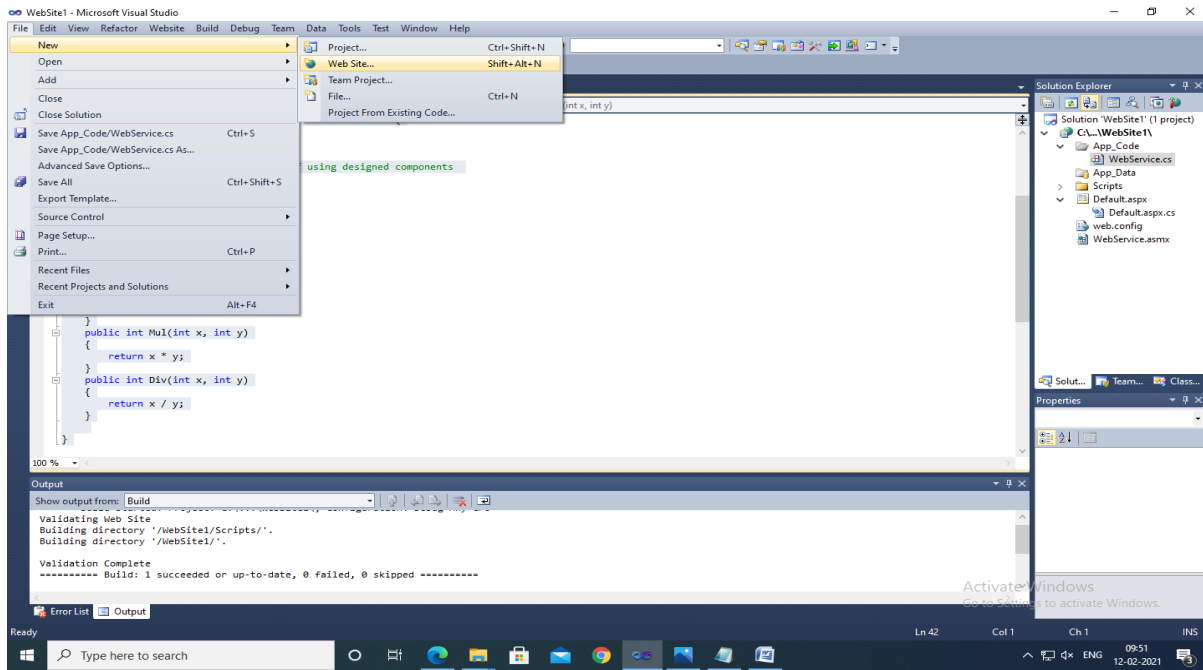
**each steps with screenshot:**



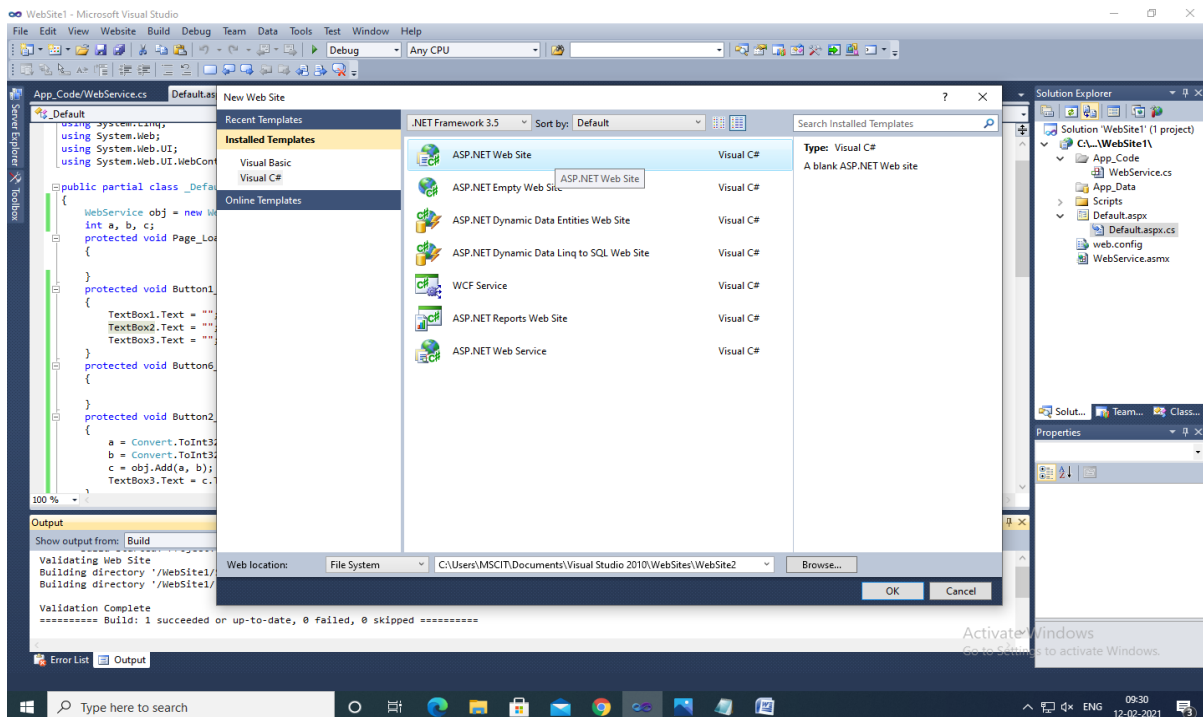
**Output:**

**Steps:**

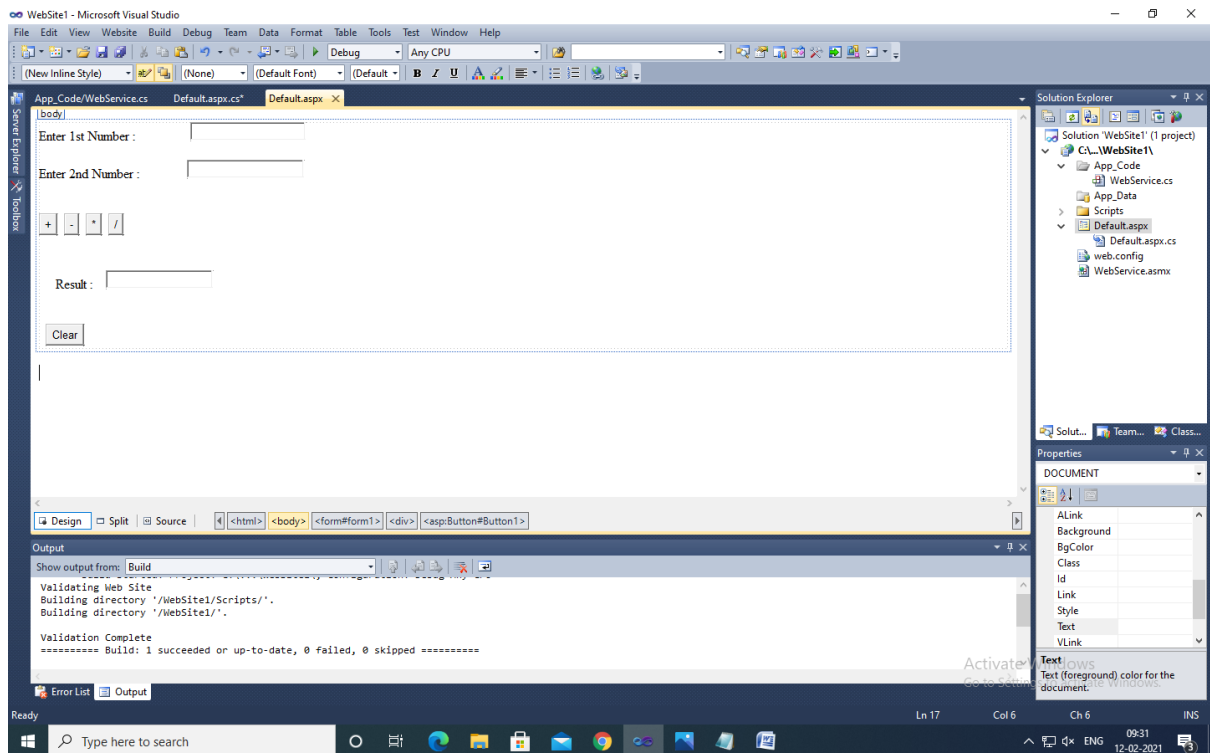
**Open visual studio 10**  
**File – New – web site Click.**



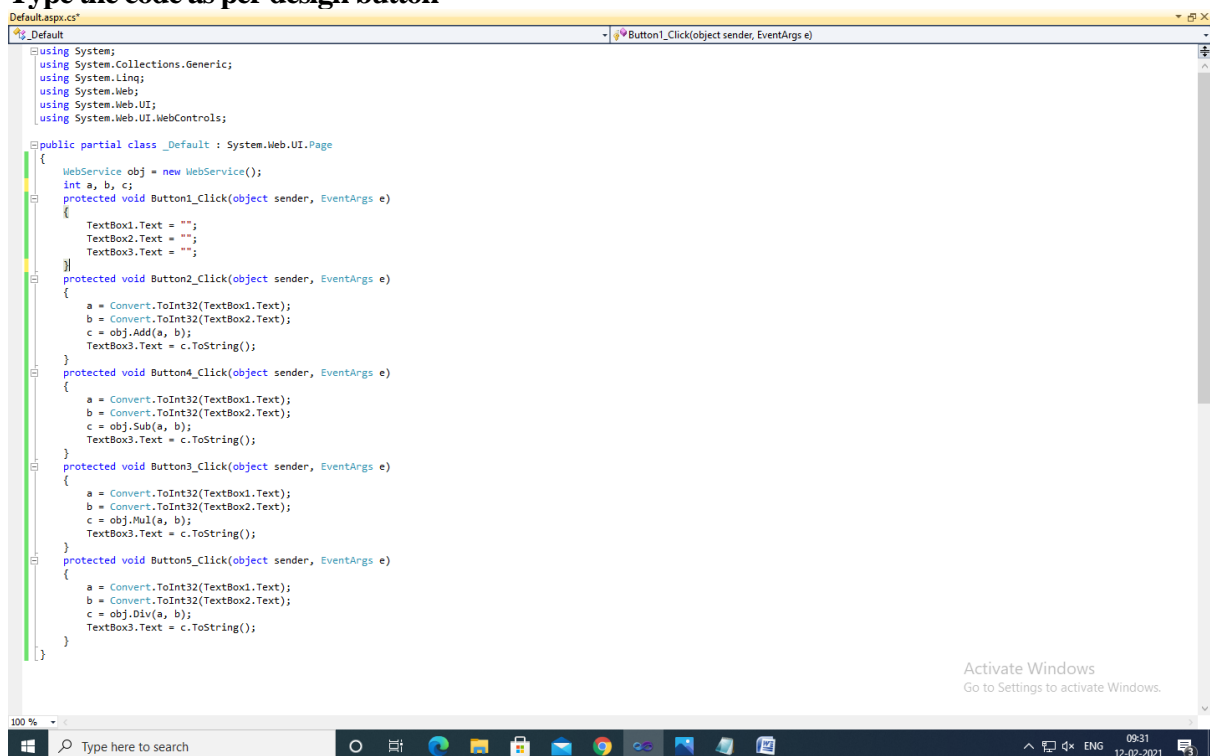
**Select ASP.NET Web Site and click ok.**



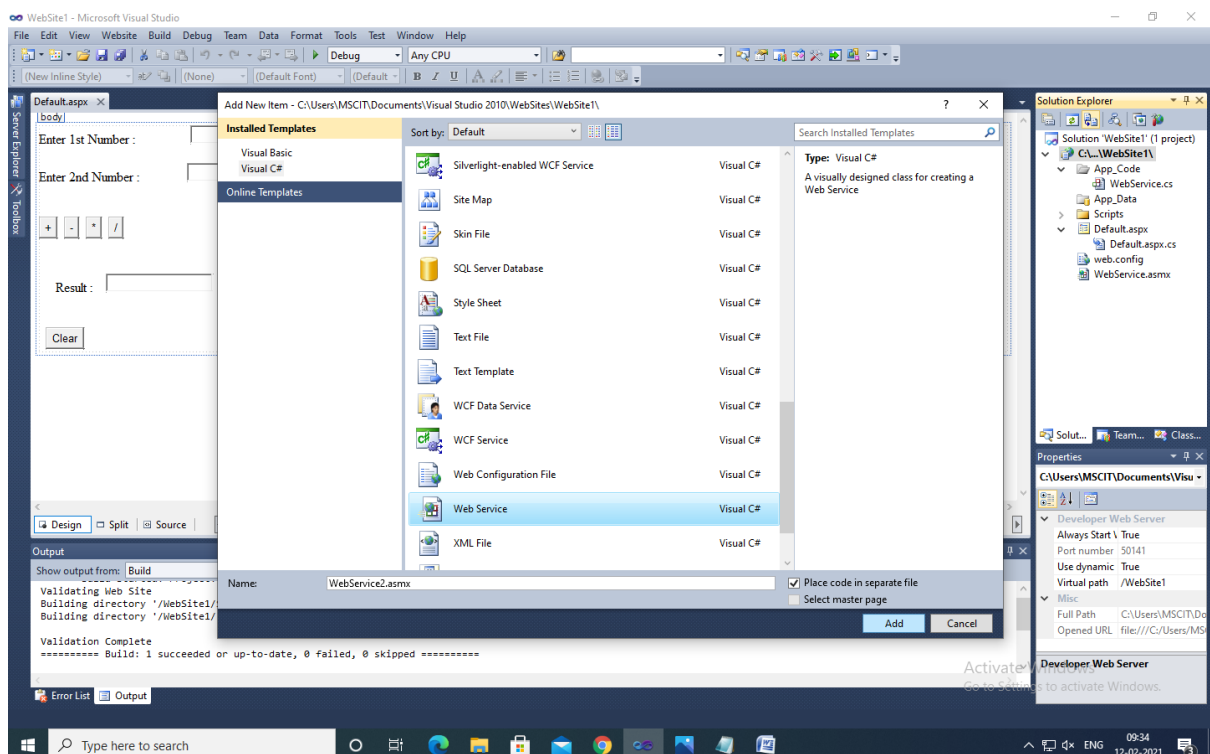
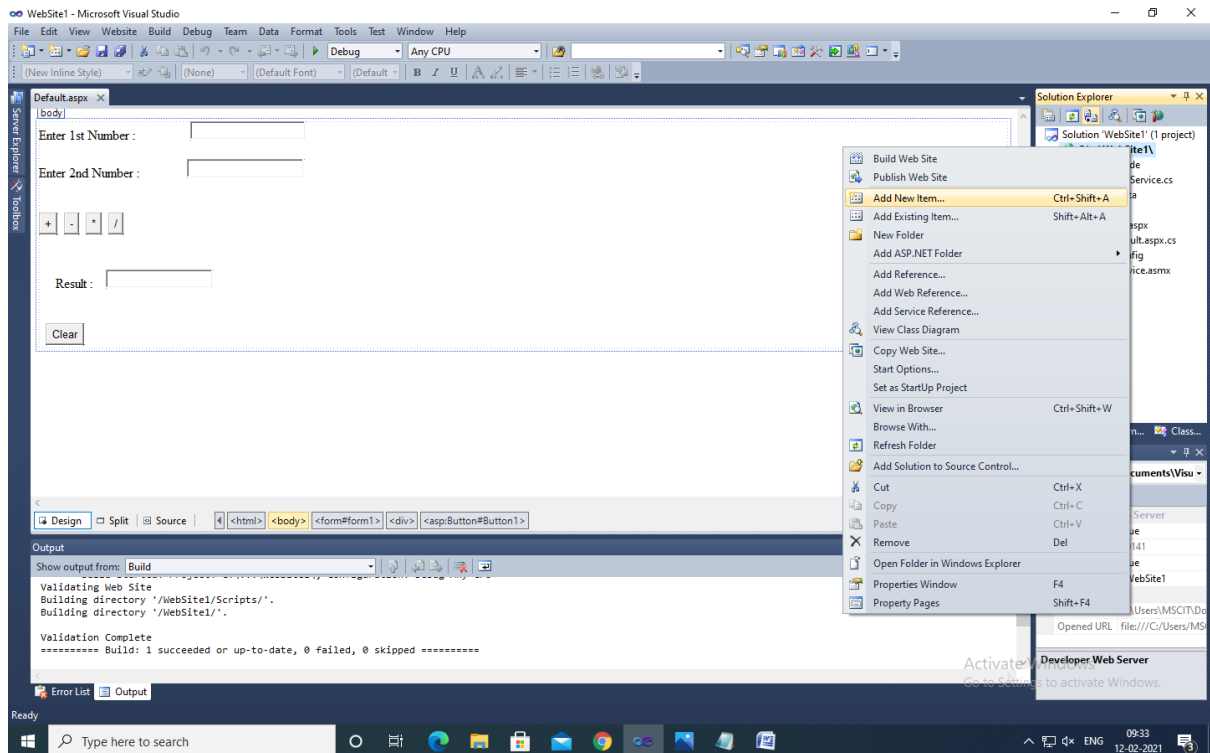
**Then create design**



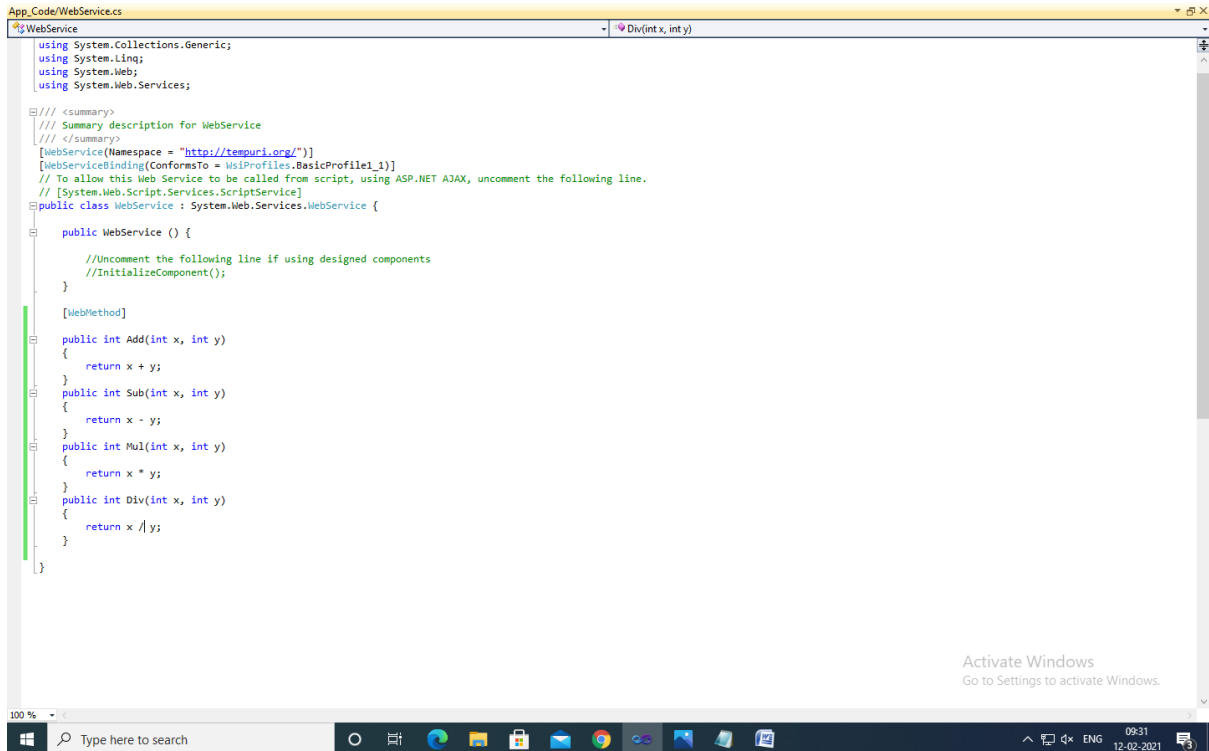
## Type the code as per design button



**Right click on website – add New item – web service - add**



## Type the code



```
App_Code/WebService.cs
WebService
Div(int x, int y)

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

/// <summary>
/// Summary description for WebService
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsProfiles.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]

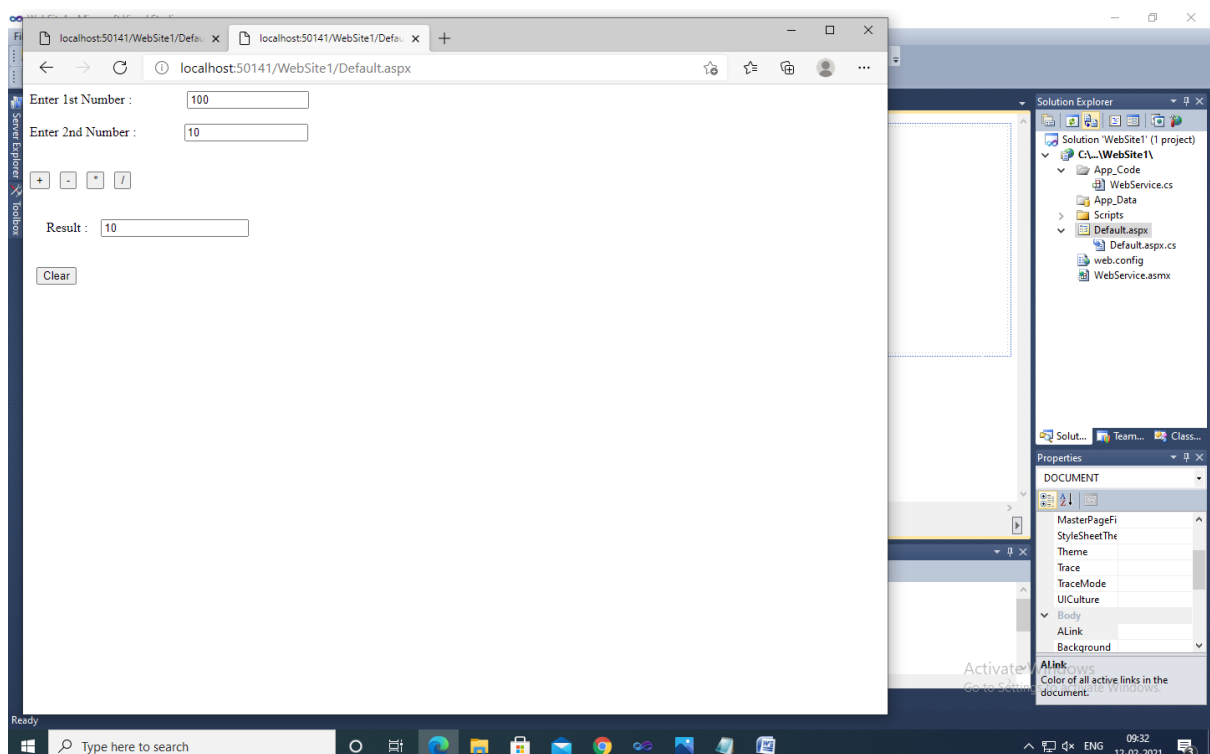
public class WebService : System.Web.Services.WebService {

    public WebService () {
        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

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

Activate Windows  
Go to Settings to activate Windows.

## And run



## **Practical No: 05**

**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.

32

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

**Code:**  
**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:

```
C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>javac TokenServer.java

C:\Neha Dawale MscIT(Sem I)>java TokenServer
The message is ClientOne.....hii
The message is ClientOne.....hello
The message is ClientTwo.....how r u
The message is ClientTwo.....good

C:\Neha Dawale MscIT(Sem I)>
```

```
C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>javac TokenClient1.java

C:\Neha Dawale MscIT(Sem I)>java TokenClient1
Do you want to enter the Data -->YES/NO
yes
ready to send
Enter the Data
hii
now sending.....
Data Sent
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
no

C:\Neha Dawale MscIT(Sem I)>_
```

```
C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>javac TokenClient2.java

C:\Neha Dawale MscIT(Sem I)>java TokenClient2
entering recieving mode
The data is Token
Do you want to enter the Data -->YES/NO
yes
ready to send
Enter the Data
how r u
now sending.....
Data Sent
Do you want to enter the Data -->YES/NO
yes
ready to send
Enter the Data
good
now sending.....
Data Sent
Do you want to enter the Data -->YES/NO
no

C:\Neha Dawale MscIT(Sem I)>_
```



## **Practical No: 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.

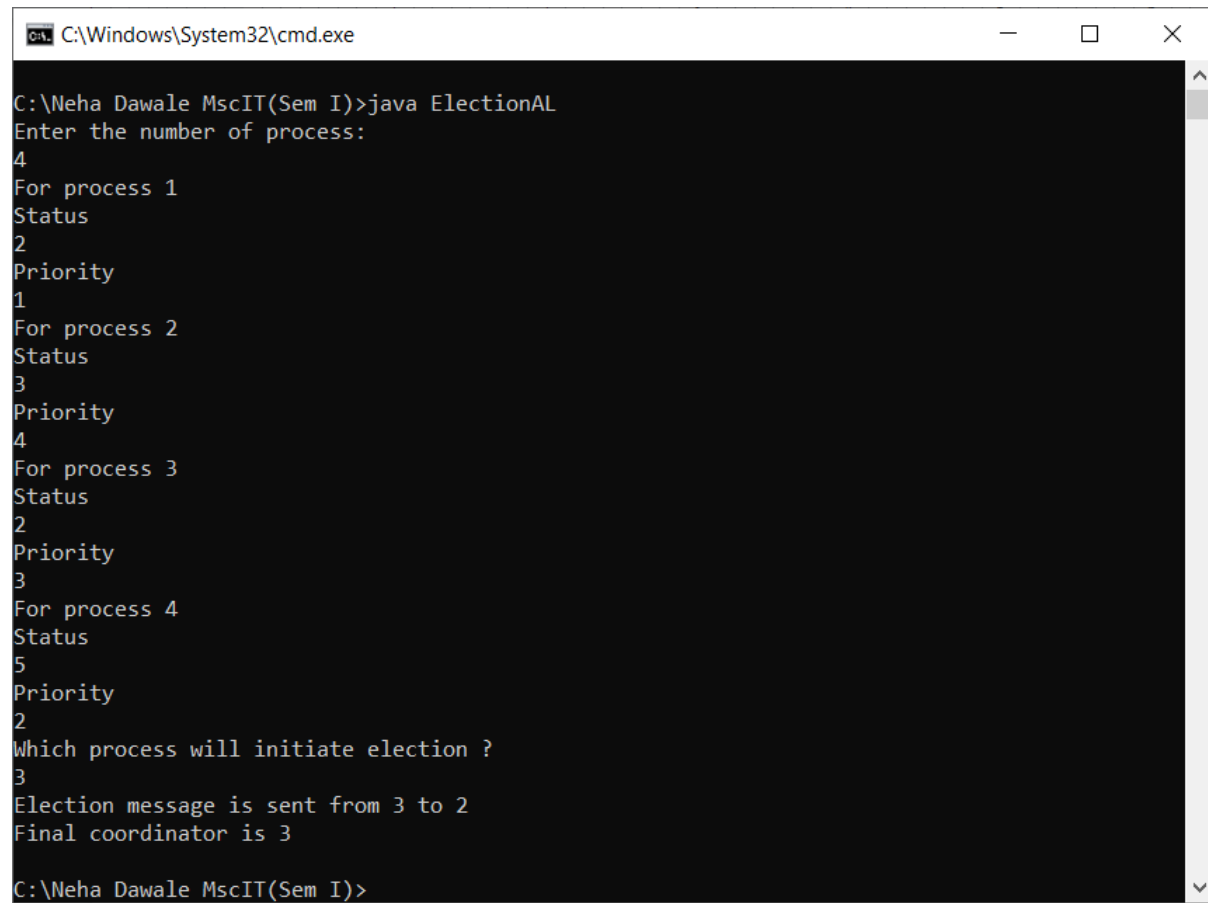
If two elections are held simultaneously (say because two different processes notice simultaneously that the coordinator is dead) then each comes up with the same list and elects the same coordinator. Some time is wasted, but nothing is really hurt by this.

## Code:

```
import java.io.*;
import java.util.Scanner;
class ElectionAL{
    static int n,co;
    static int pro[] = new int[100],sta[] = new int[100];
    public static void main(String[]Args){
        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 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:



```
C:\Windows\System32\cmd.exe
C:\Neha Dawale MscIT(Sem I)>java ElectionAL
Enter the number of process:
4
For process 1
Status
2
Priority
1
For process 2
Status
3
Priority
4
For process 3
Status
2
Priority
3
For process 4
Status
5
Priority
2
Which process will initiate election ?
3
Election message is sent from 3 to 2
Final coordinator is 3
C:\Neha Dawale MscIT(Sem I)>
```

## **Practical No: 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**

**Code:**

**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 receive;
            DateFormat dateFormat = new SimpleDateFormat("hh:mm:ss.SSSS");
            while((receive = in.readLine()) != null){
                String[] mesaage = receive.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: "+message[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 receiving time is same:"+strDate);
        }
        else if(requiretime > 2){
            String clinttim = dateFormat.format(clienttime);
            System.out.println("Message sending time from client:"+ clinttim+"\n");
            String strDate = dateFormat.format(timeMilli);
            System.out.println("Message received from client to:"+strDate+"\n");
            System.out.println("This message is rejected.");
        }
        else{
            String clinttim = dateFormat.format(clienttime);
            System.out.println("Message sending time from client:"+ clinttim+"\n");
            String strDate = dateFormat.format(timeMilli);
            System.out.println("Message 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();
        }
    }
}

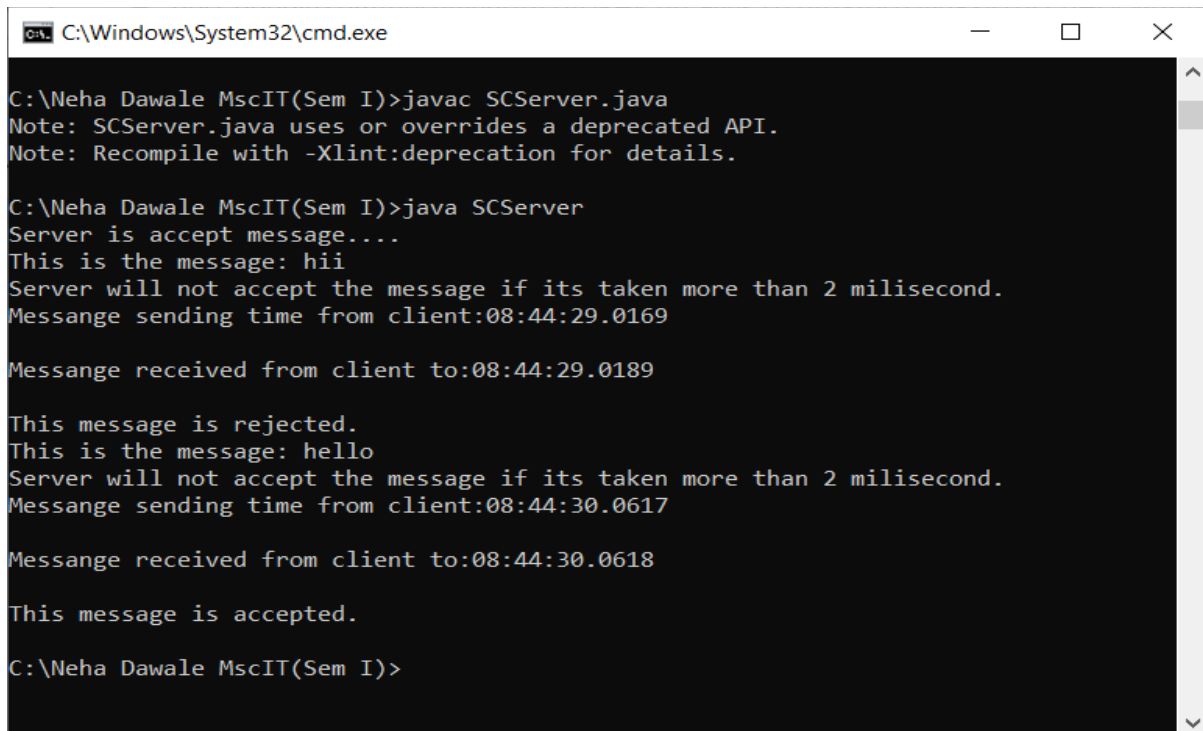
```

```

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

```

## Output:



```

C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>javac SCServer.java
Note: SCServer.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.

C:\Neha Dawale MscIT(Sem I)>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 from client:08:44:29.0169

Message received from client to:08:44:29.0189

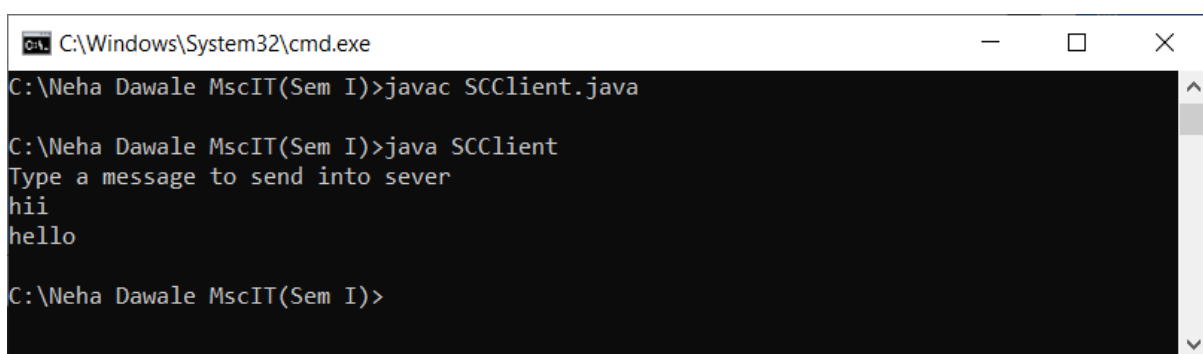
This message is rejected.
This is the message: hello
Server will not accept the message if its taken more than 2 milisecond.
Message sending time from client:08:44:30.0617

Message received from client to:08:44:30.0618

This message is accepted.

C:\Neha Dawale MscIT(Sem I)>

```



```

C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>javac SCClient.java

C:\Neha Dawale MscIT(Sem I)>java SCClient
Type a message to send into sever
hii
hello

C:\Neha Dawale MscIT(Sem I)>

```

## Practical 8

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

### **Two-Phase Commit Protocol**

Two-Phase Commit Protocol follows following steps;

Step1: The Coordinator sends a VOTE\_REQUEST message to all participants.

Step2: When a participant receives a VOTE\_REQUEST message, it returns either a VOTE\_COMMIT message to the coordinator telling the coordinator that it is prepared to commit or a VOTE\_ABORT message.

Step3: The coordinator collects all votes from the participants. If all participants had voted to commit the transaction then it sends GLOBAL\_COMMIT message to all participants. Even if one participant had voted to abort the transaction then it sends GLOBAL\_ABORT message.

Step4: Each participant that voted for a commit waits for the final reaction by the coordinator. If a participant receives a GLOBAL\_COMMIT message it locally commits the transaction else it aborts the local transaction.

### **Code:** **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 client 1
        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); //sendport=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.java**

```

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); //recport=8000
clnt.setRecPort(8000); //sendport=9001
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(),localhost,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);
}
};

```

## **TPCClient2.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 client 2

        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(),localhost,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);
    }
};
```

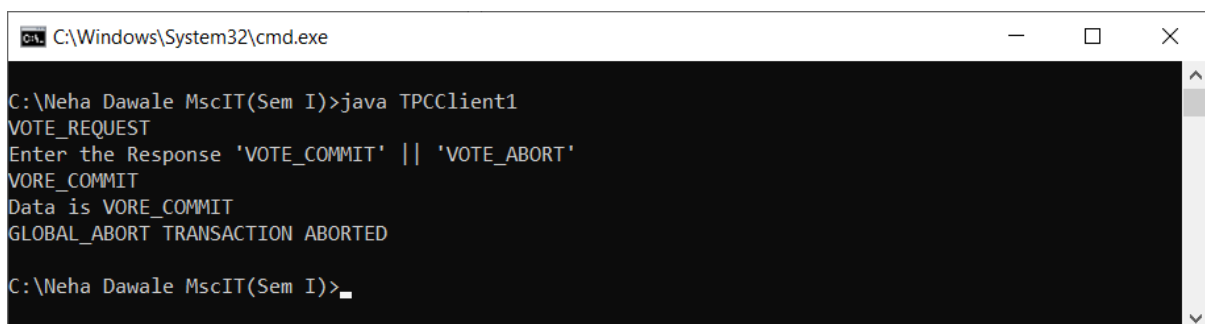
## Output:



```
C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>java TPCServer
Server in sending modeâ??..
Send request data to client1..
VOTE_REQUEST
Data is VOTE_REQUEST
Waiting for response from client1â??..
String = VORE_COMMIT
Counter value =0n   Send value =1
Send request data to client2..
VOTE_REQUEST
Data is VOTE_REQUEST
Waiting for response from client2â??..
String = VORE_COMMIT
Counter value =1n   Send value =2

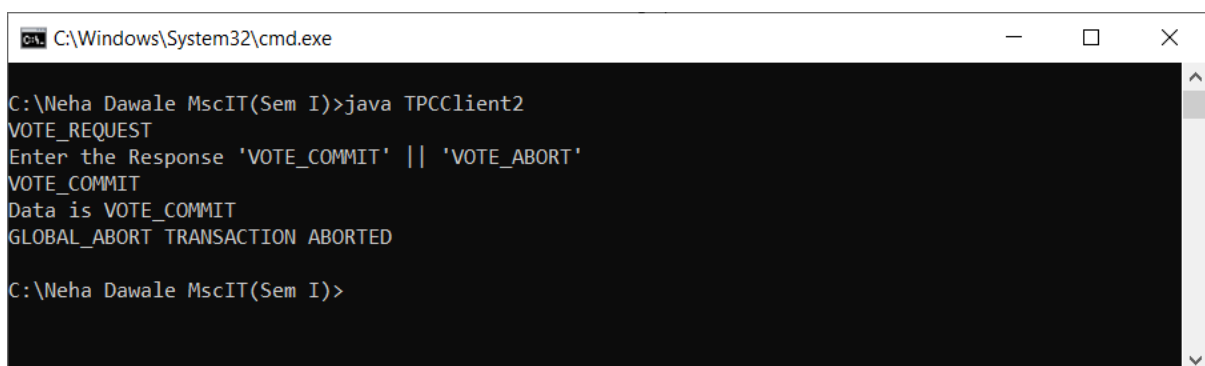
C:\Neha Dawale MscIT(Sem I)>_
```



```
C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>java TPCClient1
VOTE_REQUEST
Enter the Response 'VOTE_COMMIT' || 'VOTE_ABORT'
VORE_COMMIT
Data is VORE_COMMIT
GLOBAL_ABORT TRANSACTION ABORTED

C:\Neha Dawale MscIT(Sem I)>_
```



```
C:\Windows\System32\cmd.exe

C:\Neha Dawale MscIT(Sem I)>java TPCClient2
VOTE_REQUEST
Enter the Response 'VOTE_COMMIT' || 'VOTE_ABORT'
VOTE_COMMIT
Data is VOTE_COMMIT
GLOBAL_ABORT TRANSACTION ABORTED

C:\Neha Dawale MscIT(Sem I)>
```