**MAHARASHTRA COLLEGE OF ARTS,SCIENCE & COMMERCE**
**MUMBAI**
**400-008**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

**MSC I.T PART-1**

## CERTIFICATE

This is to certify that the practical done at MAHARASHTRA COLLEGE By Ms/Mr._____

(Seat No._____) in partial fulfillment for M.Sc.Degree Examination has been found satisfactory. This practical journal had not been submitted for any other examination and does not form part of any other course undergone by candidate.

| | | |
|---|---|---|
| Signature | Signature | Signature |
| Lecture-In-Charge | External Examiner | Course Coordinator |
| Guided by | Examined By | Certified By |

**College Stamp**

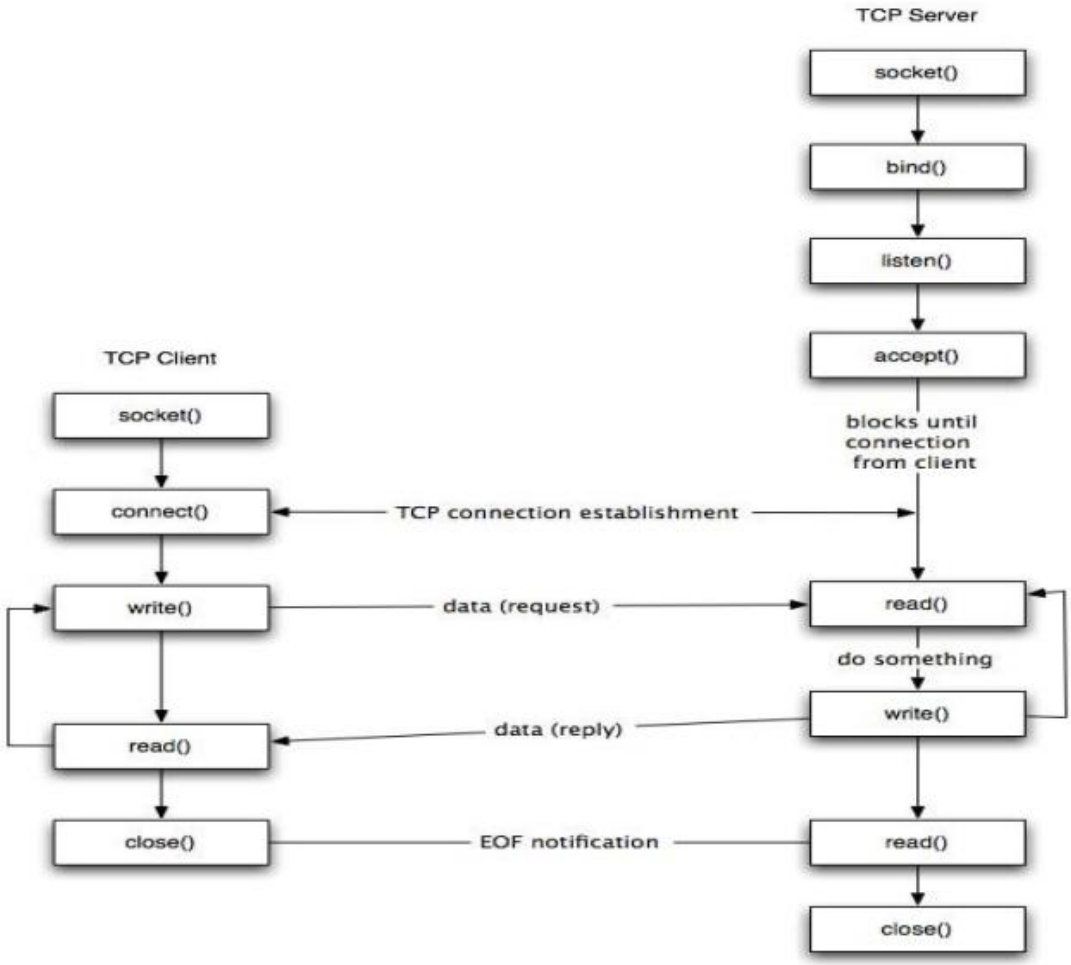
# Cloud Computing

# INDEX

## Practical No.1

**TCP (Transmission control protocol)**

A TCP (transmission control protocol) is a connection-oriented communication. It is an intermediate layer of the application layer and internet protocol layer in OSI model. TCP is designed to send the data packets over the network. It ensures that data is delivered to the correct destination.TCP creates a connection between the source and destination node before transmitting the data and keep the connection alive until the communication is active.




# Practical No: 01

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

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

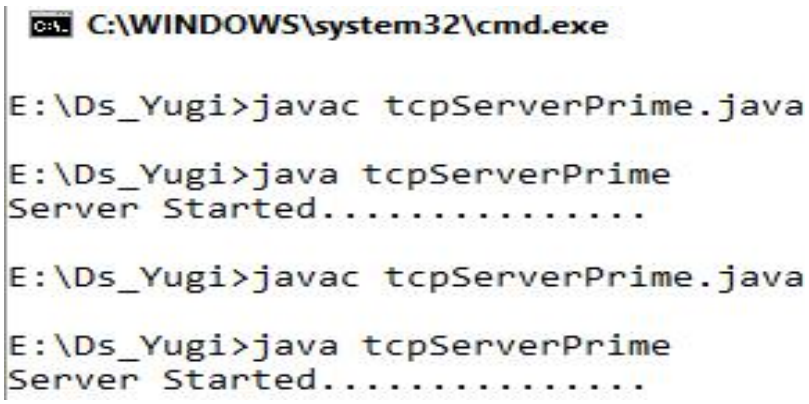
# Code:

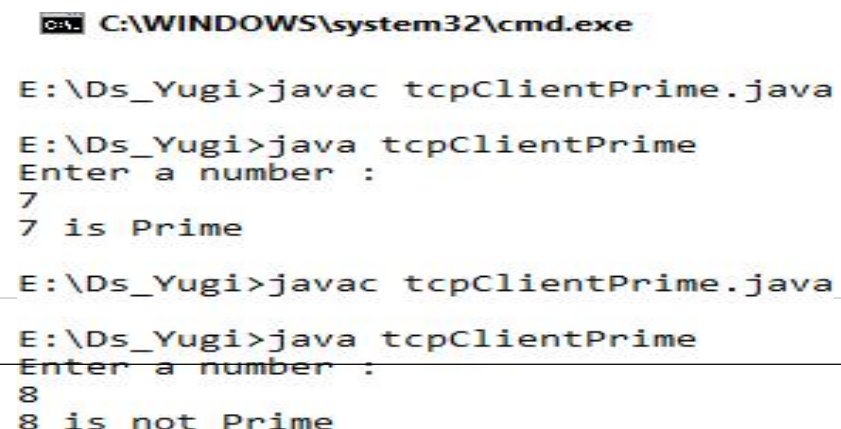
## 1. tcpServerPrime.java

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

class tcpServerPrime

{
	public static void main(String args[])
	{
		try
		{
			ServerSocket ss = new ServerSocket(8001);
			System.out.println("ServerStarted.....");
			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 || x ==2 || x ==3)
			{
				otc.writeUTF(x + "is Prime");
				System.exit(0);
			}
			for(int i=2; i<=y; i++)
			{
				if(x%i != 0)
				{
					otc.writeUTF(x + " is Prime");
				}
				else
				{
					otc.writeUTF(x + " is not Prime");
				}

			}
		}
		catch(Exception e)

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

## 1. tcpClientPrime.java

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

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

   Socket cs = new Socket("LocalHost",8001); BufferedReader infu =
   new BufferedReader(new InputStreamReader(System.in));

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

                        int a = Integer.parseInt(infu.readLine());
                        DataOutputStream out = new
DataOutputStream(cs.getOutputStream());
                        out.writeInt(a);

                        DataInputStream in = new DataInputStream(cs.getInputStream());
                        System.out.println(in.readUTF()); cs.close();

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



## 1E. A Client-Server TCP based chatting Application.

### Program Code:

**Server**

```
package tcpchatapp_server;

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

public class TCPChatApp_Server
{

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

            int port = 7777;
            ServerSocket ss = new ServerSocket(port);
            System.out.println("Waiting for client....");
            Socket s = ss.accept();
            System.out.println("Connection is done....");

            while(true)
            {
                BufferedReader br1 = new BufferedReader(new InputStreamReader(s.getInputStream()));
                String ctext = br1.readLine();
                System.out.println("Client text :" +ctext);

                BufferedReader br2 = new BufferedReader(new InputStreamReader(System.in));
                System.out.println("Enter Text:  ");
                String stext = br2.readLine();

                PrintWriter pw = new PrintWriter(new OutputStreamWriter(s.getOutputStream()));
                pw.println(stext);
                pw.flush();

                if (stext.equalsIgnoreCase("bye"))
                {
                    System.exit(0);
                }
            }
        }

       catch (Exception e)
        {
            System.out.println("Error" + e);
        }
    }
}
```

# Client

```
package tcpchatapp_client;

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

public class TCPChatApp_Client
{
    public static void main(String[] args)
    {
      try
        {
           System.out.println("```Name:- OWAIS AHMAD```");
           int port = 7777;
           InetAddress addr = InetAddress.getLocalHost();
           Socket s = new Socket(addr,port);
           System.out.println("Connection is done");

           while(true)
            {
               BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
               System.out.println("Enter Text: ");
               String ctext = br1.readLine();

               PrintWriter pw = new PrintWriter(new OutputStreamWriter(s.getOutputStream()));
               pw.println(ctext);
              pw.flush();

             if(ctext.equalsIgnoreCase("bye"))
               {
                 System.exit(0);
               }

               BufferedReader br2 = new BufferedReader(new InputStreamReader(s.getInputStream()));
               String stext = br2.readLine();
               System.out.println("Server Text:  " +stext);
            }
        }

      catch (Exception e)
        {
          System.out.println("Error" + e);
        }
    }
}
```

## Output:

### 1) Server Output-

```
Waiting for client....
Connection is done....
Client text :Hi!
Enter Text:
Hello
Client text :How are you?
Enter Text:
All good.
Client text :Bye
Enter Text:
Bye
BUILD SUCCESSFUL (total time: 1 minute 43 seconds)
```

### 2) Client Output-

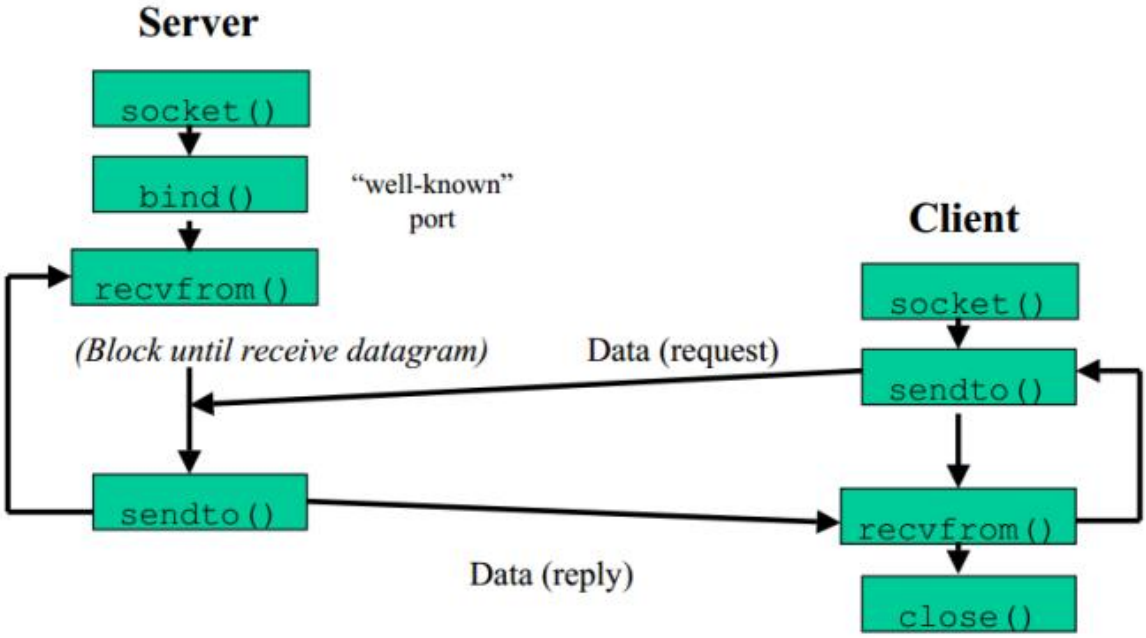
```
Connection is done
Enter Text:
Hi!
Server Text:  Hello
Enter Text:
How are you?
Server Text:  All good.
Enter Text:
Bye
BUILD SUCCESSFUL (total time: 1 minute 14 seconds)
```

## Practical No.2

**User Datagram Protocol**

User Datagram Protocol (UDP) is a fundamental network protocol for transmission of messages referred to as datagram. UDP has minimal overheads. There are no acknowledgments and, therefore, delivery is not guaranteed. Datagram may be delivered out of sequence. However, packets have a checksum and data integrity of a packet is guaranteed. Because of low overheads, UDP is fast and suitable in many situations, particularly in real time systems. For example, if some periodic data is being transmitted, a few lost packets might not matter because similar relevant and more current data is coming next at short intervals.

# UDP Client-Server



# Practical-2

## 2A. A Client-Server based program using UDP to find if the number entered is odd or even.

## Program Code:

### Server

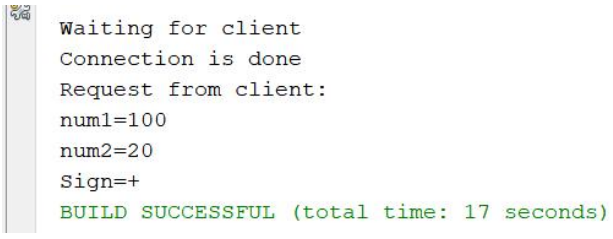
```
package udpoddeven_server;

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

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

            int port= 7777;
            DatagramSocket ds = new DatagramSocket(port);
            System.out.println("Waiting for client");

            while(true)
             {
                byte b[]= new byte[1024];
                DatagramPacket dp1 = new DatagramPacket(b,b.length);
                ds.receive(dp1);

                String str = new String(dp1.getData(),0,dp1.getLength());
                if(str.equals("end"))
                {
                    System.exit(0);
                }
               int num = Integer.parseInt(str);

               System.out.println("Request from client: "+ num);
               Str
               if(
                {
                }

               byt
               b1
               Ine
               Dat

               ds.
             }
        }

      catch (Except
        {

            System.
        }
    }
}
```

### Client

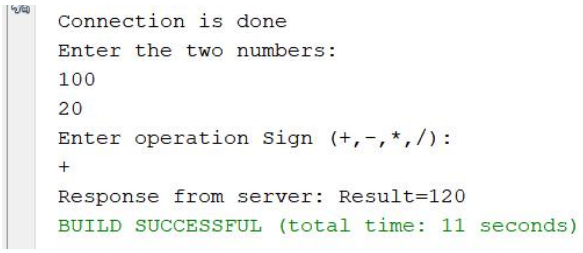
```
package tcpcalci_client;

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

public class TCPCalci_Client {
    public static void main(String[] args)
    {
       try
        {
            System.out.println("```OWAIS AHMAD```");
            int port = 8888;
            InetAddress addr = InetAddress.getLocalHost();
            Socket s = new Socket(addr,port);
            System.out.println("Connection is done");

            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter the two numbers: ");
            int n1 = Integer.parseInt(br.readLine());
            int n2 = Integer.parseInt(br.readLine());
            System.out.println("Enter operation Sign (+,-,*,/): ");
            String sign = br.readLine();

            PrintWriter pw = new PrintWriter(new OutputStreamWriter(s.getOutputStream()));
            pw.println(n1);
            pw.println(n2);
            pw.println(sign);
            pw.flush();

            BufferedReader br1 = new BufferedReader(new InputStreamReader(s.getInputStream()));
            String res = br1.readLine();
            System.out.println("Response from server: " +res);
        }
       catch (Exception e)
        {
          System.out.println("Error" +e);
        }
    }
}
```

## Output:

1) **Output for Addition-**



```
Waiting for client
Connection is done
Request from client:
num1=100
num2=20
Sign=+
BUILD SUCCESSFUL (total time: 17 seconds)
```



```
Connection is done
Enter the two numbers:
100
20
Enter operation Sign (+,-,*,/):
+
Response from server: Result=120
BUILD SUCCESSFUL (total time: 11 seconds)
```

**2B. A Client-Server based program using UDP to find the factorial of a number.**

## Program Code:

## Server

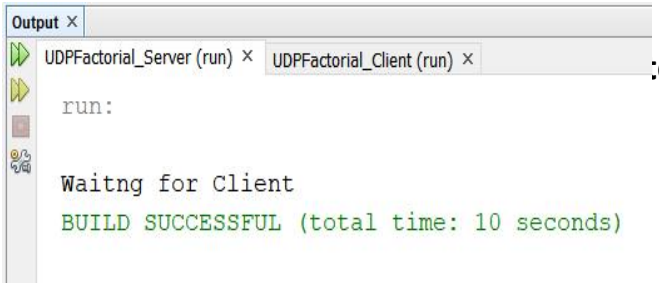
```
package udpfactorial_server;

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

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

            int port = 8888;
            DatagramSocket ds = new DatagramSocket(port);
            System.out.println("Waitng for Client");

            byte b[] = new byte[1024];
            DatagramPacket dp1 = new DatagramPacket(b,b.length);
            ds.receive(dp1);

            String str = new String(dp1.getData(),0,dp1.getLength());
            int num = Integer.parseInt(str);

            int f = 1;
            for(int i=1;i<=num;i++)
            {
                f
            }
            Strin

            byte
            b1 =
            InetA
            Datag
            ds.se
        }

       catch (Exc
        {
            Syste
        }
    }
}
```

## Client

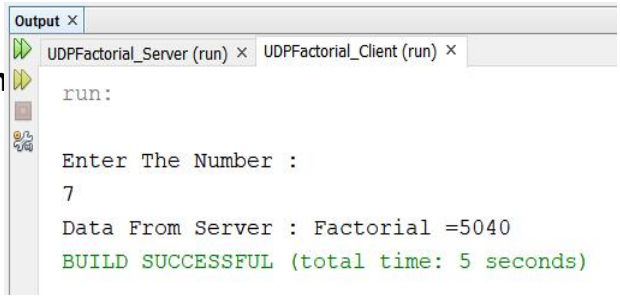
```
package udpfactorial_client;
import java.io.*;
import java.net.*;

public class UDPFactorial_Client {

    public static void main(String[] args)
    {
        try
        {
            System.out.println("```OWAIS AHMAD```");
            int port = 9999;
            DatagramSocket ds =  new DatagramSocket(port);

            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter The Number :");
            String num = br.readLine();

            byte b[] = new byte[1024];
            b = num.getBytes();
            InetAddress addr = InetAddress.getLocalHost();
            DatagramPacket dp1 = new DatagramPacket(b,b.length,addr,8888);
            ds.send(dp1);

            byte b1[] = new byte[1024];
            DatagramPacket dp2 = new DatagramPacket(b1,b1.length);
            ds.receive(dp2);

            String str = new String(dp2.getData(),0,dp2.getLength());
            System.out.println("Data From Server : " + str);
        }

        catch (Exception e)

            System.out.println("Error :" +e);
        }
    }
}
```

## Output:

**Server Output-**



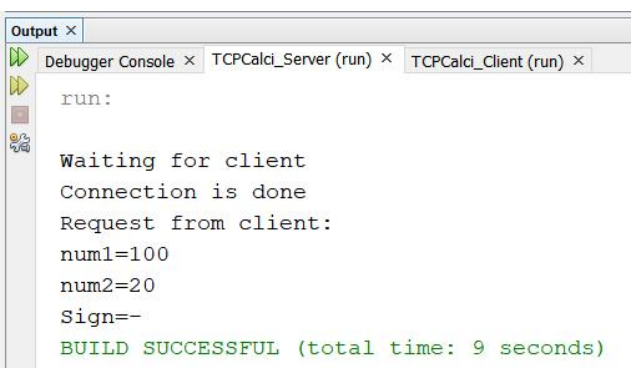

**2) Client Output-**

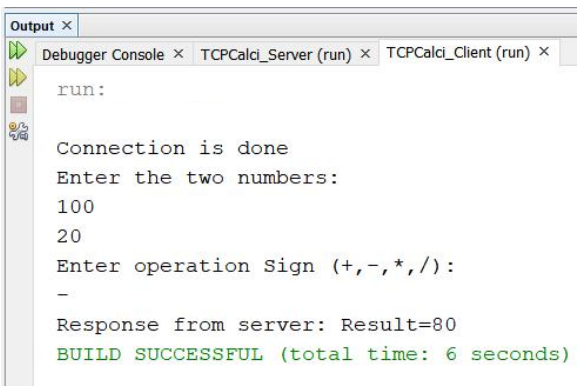


# Server

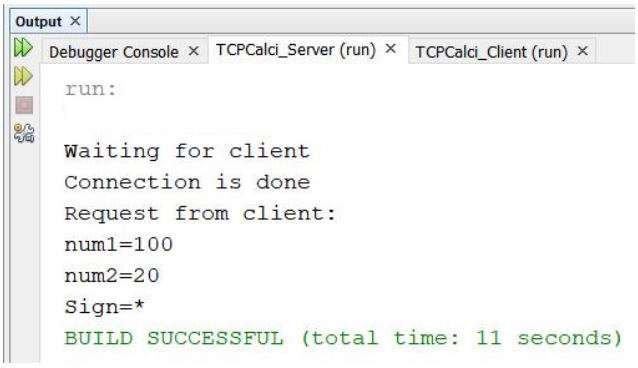
```
package tcpcalci_server;

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

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

            int port = 8888;
            ServerSocket ss = new ServerSocket(port);
            System.out.println("Waiting for client");
            Socket s = ss.accept();
            System.out.println("Connection is done");

            BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream()));
            int num1 = Integer.parseInt(br.readLine());
            int num2 = Integer.parseInt(br.readLine());
            String sign = br.readLine();

            System.out.println("Request from client: ");
            System.out.println("num1="+num1);
            System.out.println("num2="+num2);
            System.out.println("Sign="+sign);
            int res = 0;
            if (sign.equals("+"))
            {
                res = num1 + num2;
            }
            else if (sign.equals("-"))
            {
                res = num1 - num2;
            }
            else if (sign.equals("*"))
            {
                res = num1 * num2;
            }else if (sign.equals("/"))
            {
                res = num1 / num2;
            }
            String response = "Result=" +res;
            PrintWriter pw = new PrintWriter(new OutputStreamWriter(s.getOutputStream()));
            pw.println(response);
            pw.flush();
        }
        catch (Exception e)
        {
            System.out.println("Error" +e);
        }
    }
```

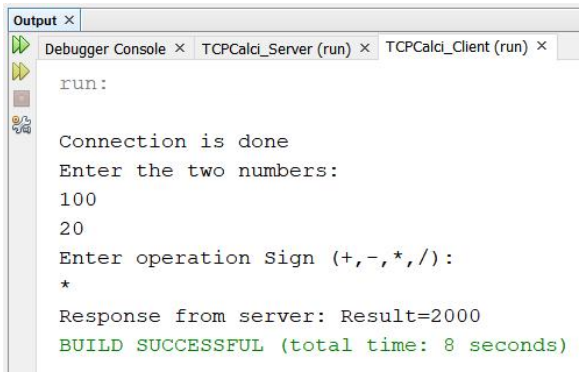
### 1) Output for Subtraction-



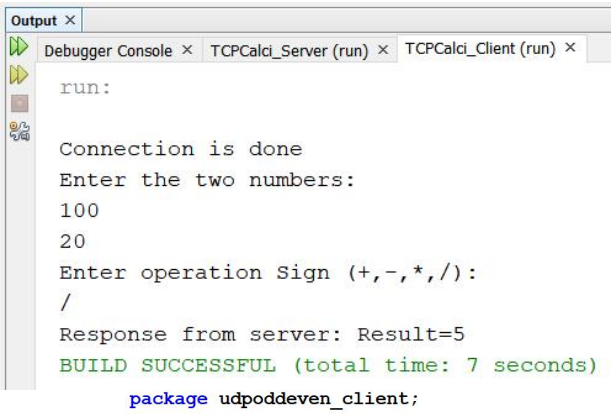



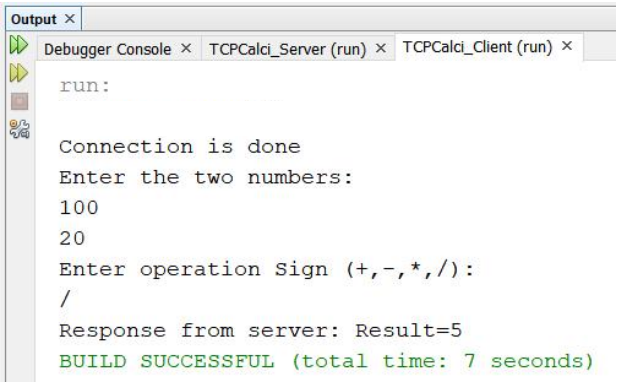
### 2) Output for Multiplication-




### 3) Output for Division-



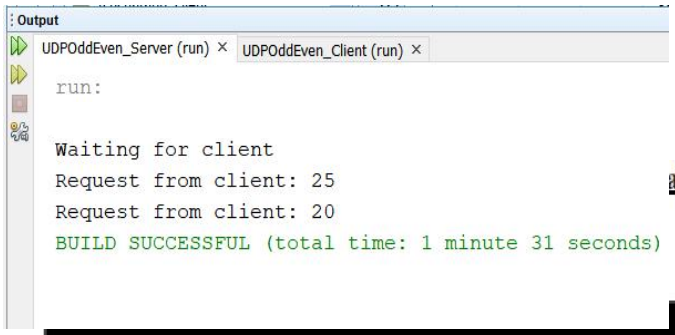




**Clie**

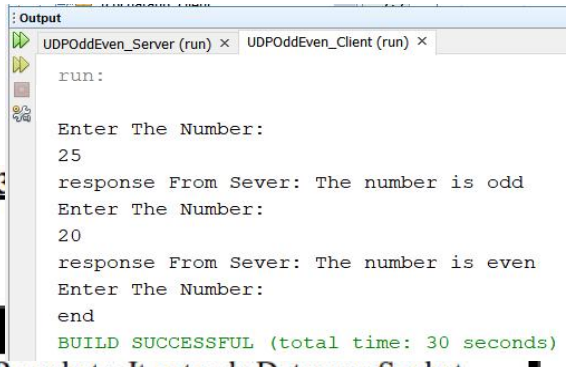
```
package udpoddeven_client;

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

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

          int port = 8888;
          DatagramSocket ds = new DatagramSocket(port);

          while(true)
           {
             BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
             System.out.println("Enter The Number: ");
             String num = br.readLine();

             byte b[] = new byte[1024];
             b = num.getBytes();
             InetAddress addr = InetAddress.getLocalHost();
             DatagramPacket dp1 = new DatagramPacket(b,b.length,addr,7777);
             ds.send(dp1);

             if(num.equals("end"))
              {
                System.exit(0);
              }

             byte b1[] = new byte[1024];
             DatagramPacket dp2 = new DatagramPacket(b1,b1.length);
             ds.receive(dp2);
             String str = new String(dp2.getData(),0,dp2.getLength());
             System.out.println("response From Sever: " +str);
             }
        }

       catch (Exception e)
       {
         System.out.println("Error "+ e.getMessage());
       }
    }
}
```

**Output:**

**Server Output-**




**2) Client Output-**




This class is used for sending and receiving multicast IP packets. It extends DatagramSocket class and provides additional functionality for joining groups. A message sent to the group IP address will be received by all the clients who have joined the group. This must be kept in mind that for sending packets to the group, datagram socket doesn't have to join the group but for receiving the packets addressed to the group, it must join the group. This class provides various methods for controlling the flow of multicast packets like setting the ttl, network interface to use etc, along with the major functions of joining and leaving a group.

Constructors :

1. public MulticastSocket() : Creates a multicast socket. When using this constructor, we have to explicitly set all the fields such as group address, port number etc.

Syntax :public MulticastSocket()

2. public MulticastSocket(int port) : Creates a multicast socket bound on the port specified.

Syntax :public MulticastSocket(int port)

Parameters :

---

port : port number to bind this socket

# Practical-3

### 3. A Client-Server based program for multicast message using UDP.

## Program Code:

### Server

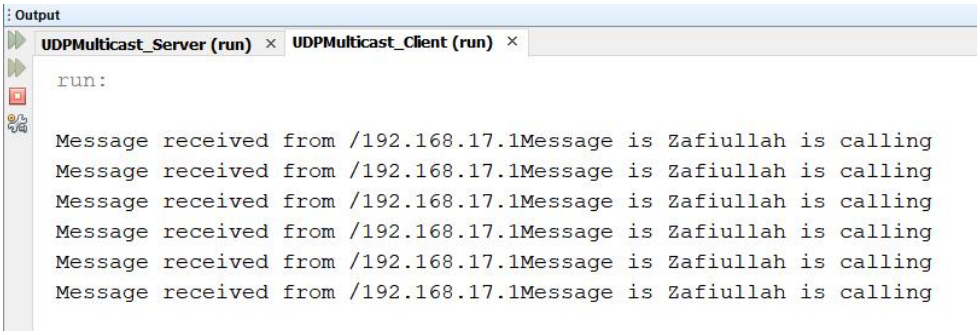
```
package udpmulticast_server;

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

public class UDPMulticast_Server {
    public static void main(String[] args) throws IOException, InterruptedException
    {
        try
        {
            System.out.println("```OWAIS AHMAD```");
            int port = 1234;
            MulticastSocket socket = new MulticastSocket();
            byte [] data = new byte [100];
            InetAddress address = InetAddress.getByName("239.1.2.3");
            socket.joinGroup(address);

            for(;;)
            {
              Thread.sleep(10000);
              System.out.println("Sending");
              String str = ("Zafiullah is calling");
              data = str.getBytes();
              DatagramPacket packet = new DatagramPacket(data,str.length(),address,port);
              socket.send(packet);
            }
        }

        catch (Exception e)
        {
            System.out.println("Error "+ e.getMessage());
        }
    }
}
```

## Output:

# Client

```
package udpmulticast_client;

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

public class UDPMulticast_Client {
    public static void main(String[] args)
    {
        try
        {
          System.out.println("```OWAIS AHMAD```");
          int port = 1234;
          MulticastSocket socket = new MulticastSocket(port);
          byte [] data= new byte[100];
          InetAddress address = InetAddress.getByName("239.1.2.3");
          DatagramPacket packet = new DatagramPacket(data,data.length);
          socket.joinGroup(address);

          for(;;)
           {
             socket.receive(packet);
             String str = new String(packet.getData(),0,packet.getLength());
             System.out.println("Message received from " +packet.getAddress()+"Message is " +str);
           }
        }
        catch (Exception e)
        {
            System.out.println("Error "+ e.getMessage());
        }
    }
}
```

## Output:



## Practical 2D:

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

## Code:

1. **RPCNumServer.java**

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


{
        DatagramSocketds;

        DatagramPacketdp;
        String str,methodName,result;
        int val;
        RPCNumServer()
        {
                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;
                                        val = Integer.parseInt(st.nextToken());
                                }
                        }
                        System.out.println(str);
                        InetAddress ia = InetAddress.getLocalHost();
                        if(methodName.equalsIgnoreCase("square"))
                        {
                                result= "" + square(val);
                        }
                        else if(methodName.equalsIgnoreCase("squareroot"))
                        {
                                result= "" + squareroot(val);
                        }
                        else if(methodName.equalsIgnoreCase("cube"))
                        {
                              result= "" + cube(val);

                        }
                        else if(methodName.equalsIgnoreCase("cuberoot"))
                        {
                                result= "" + cuberoot(val);
                        }
                        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 double square(int a) throws Exception
        {
              double ans; ans = a*a;
              return ans;
        }
        public double squareroot(int a) throws Exception
        {
              double ans;
              ans = Math.sqrt(a);
              return ans;
        }
         public double cube(int a) throws Exception
```

```
        {
                double ans; ans = a*a*a;
                return ans;
        }

        Public double cuberoot(int a) throws Exception
        {
                double ans;
                ans = Math.cbrt(a); return ans;
        }
        public static void main(String[] args)
        {
                new RPCNumServer();
        }

}
```

1. **RPCNumClient.java**

```
import
java.io.*;
import
java.net.*;
class
RPCNumClien
t
{
        RPCNumClient()
        {
                try


                {

                        InetAddress ia = InetAddress.getLocalHost();
                        DatagramSocket ds = new DatagramSocket();
                        DatagramSocket ds1 = new
                        DatagramSocket(1300);
                        System.out.println("\nRPC Client\n");

                        System.out.println("1. Square of the number - square\n2. Square root
of the number - squareroot\n3. Cube of the number - cube\n4. Cube root of the
number - cuberoot");

                        System.out.println("Enter method name and the
                        number\n");




                        while (true)
```

```
                        {
                         BufferReader br = new BufferReader(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();
                }
        }
        public static void main(String[] args)
        {
                new RPCNumClient();
        }
}
```

## Output:

```
C:\WINDOWS\system32\cmd.exe - java RPCNumServer

E:\Ds_Yugi>javac RPCNumServer.java

E:\Ds_Yugi>java RPCNumServer
square 7
result : 49.0

squareroot 25
result : 5.0

cube 3
result : 27.0

cuberoot 27
result : 3.0
```

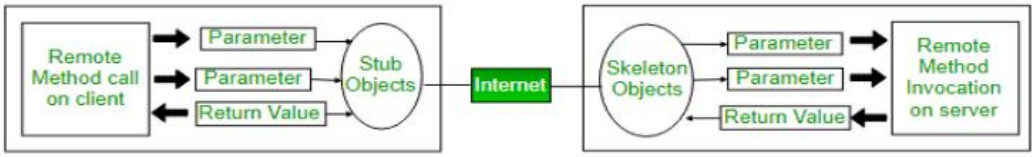
```
C:\WINDOWS\system32\cmd.exe - java RPCNumClient

1. Square of the number - square
2. Square root of the number - squareroot
3. Cube of the number - cube
4. Cube root of the number - cuberoot
Enter method name and the number

square 7

Result = 49.0

squareroot 25

Result = 5.0

cube 3

Result = 27.0

cuberoot 27

Result = 3.0
```

# Practical No.4

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*.




Working of RMI

The is given the 6 steps to write the RMI program.

1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmic tool
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

# Practical-4

## 4A. A RMI based application program to display Current Date and time.

1. **Create four files:**
   a. Interface file
   b. Implementation file
   c. Server file
   d. Client file

2. **Import three Packages:**
   i. import java.rmi.*;
   ii. import java.rmi.server.*;
   iii. import java.io.*;

### a. Interface file –

```
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
public interface DateTimeInter extends Remote
{
        String getDateTime()throws Exception;
}
```

### b. Implementation File –

```
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
public class DateTimeImpl extends UnicastRemoteObject implements DateTimeInter
{
        public DateTimeImpl()throws Exception
        {
                super();
        }
        public String getDateTime()throws Exception
        {
                Date d = new Date();
                return d.toString();
        }

}
```

### c. Server file –

```
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
public class DateTimeServer
{
        public static void main(String args[])throws Exception
        {
                DateTimeImpl obj = new DateTimeImpl();
                Naming.bind("abc",obj);
                System.out.println("Waiting for client");
        }
}
```

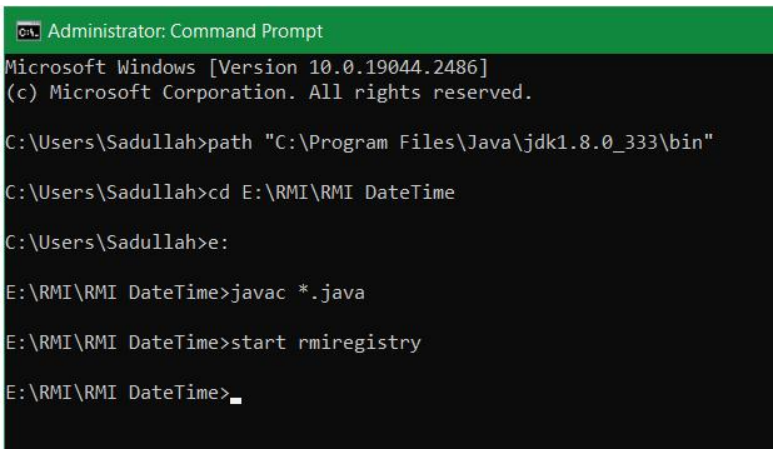
### d. Client file –

```
import java.io.*;
import java.rmi.*;
import java.rmi.server.*;
import java.util.*;
public class DateTimeClient
{
        public static void main(String args[])throws Exception
        {
                DateTimeInter obj = (DateTimeInter) Naming.lookup("abc");

                String response = obj.getDateTime();
                System.out.println("Today's Date Time is " + response);
        }
}
```

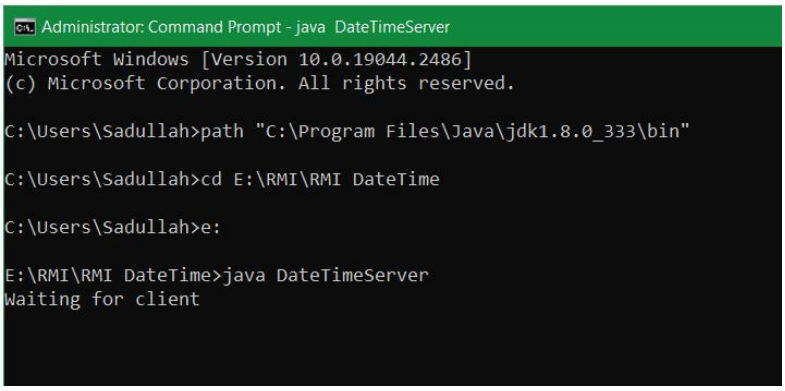
## 3. Steps to run RMI Program:

**Steps 1:** Open Command prompt
**Steps 2:** Set the 'JAVA > bin' folder path
**Steps 3:** Compile all four files
**Steps 4:** Start RMI Registry

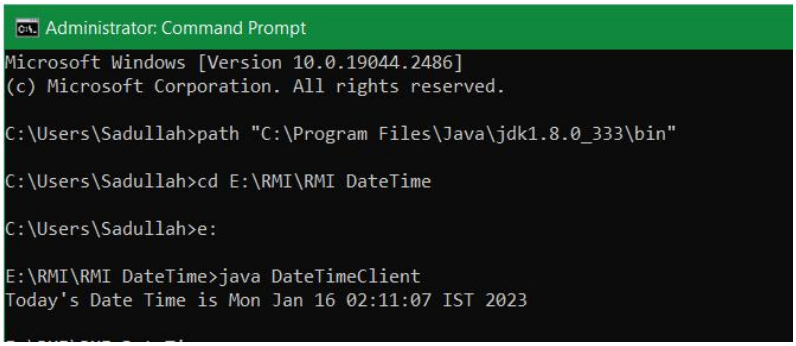
## Compile all four files & Start RMI Registry:




## Server Output:




## Client Output:



## 4B. RMI based application program that converts digits to words, e.g., 123 will be converted to one two three.

# Code:

1. **InterConvert.java**

```
import java.rmi.*;

public interface InterConvert extends Remote
{
        public String convertDigit(String no) throws Exception;
}
```

1. **ServerConvert.java**

```
import java.rmi.*;
importjava.rmi.se
rver.*;

public class ServerConvert extends UnicastRemoteObject implements
InterConvert {

        public ServerConvert() throws Exception
        {
        }

        public String convertDigit(String no) throws Exception
        {

                String str = "";
                for(int i = 0; i < no.length(); i++)
                {
                        int p =
                        no.charAt(i);
                        if( p == 48)
                        {
                                str += "zero ";
                        }
                        if( p == 49)
                        {
                                str += "one ";
                        }
                        if( p == 50)
                        {
                                str += "two ";
```

```
                        }
                        if( p == 51)
                        {
                                str += "three ";
                        }
                        if( p == 52)
                        {
                                str += "four ";


                        }
                        if( p == 53)
                        {
                                str += "five ";
                        }
                        if( p == 54)
                        {
                                str += "six ";
                        }
                        if( p == 55)
                        {



                                str += "seven ";
                        }
                        if( p == 56)
                        {

                                str += "eight ";
                        }
                        if( p == 57)
                        {
                                str += "nine ";
                        }
                }
                return str;
        }

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

                ServerConvert s1 = new
                ServerConvert();
                Naming.bind("Wrd",s1);
                System.out.println("Objectregistere
                d..... ");
        }
}
```

## 1. ClientConvert.java

```
import
java.rmi.*;
import
java.io.*;
public class ClientConvert
{

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

                InterConvert h1 =
                (InterConvert)Naming.lookup("Wrd");
                BufferedReader br = new BufferedReader(new

InputStreamReader(System.in));
                System.out.println("Enter a number :
                \t"); String no = br.readLine();

                String ans = h1.convertDigit(no);
                System.out.println("The word representation of the entered digit is : " +ans);
        }
}
```

## Output:

```
C:\WINDOWS\system32\cmd.exe - rmiregistry
E:\Ds_Yugi>javac ServerConvert.java

E:\Ds_Yugi>javac ClientConvert.java

E:\Ds_Yugi>rmic ServerConvert
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.

E:\Ds_Yugi>rmiregistry
```

```
C:\WINDOWS\system32\cmd.exe - java ServerConvert

E:\Ds_Yugi>java ServerConvert
Object registered....
```

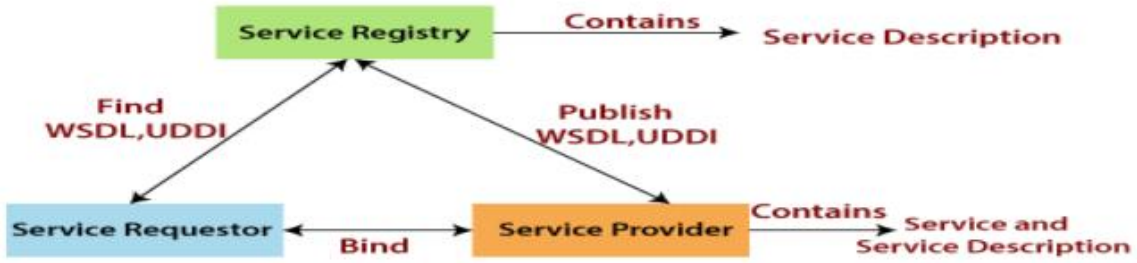
```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>java ClientConvert
Enter a number :
123
The word representation of the entered digit is : one two three

E:\Ds_Yugi>java ClientConvert
Enter a number :
456
The word representation of the entered digit is : four five six
```

## Practical No.5

The Web Services architecture describes how to instantiate the elements and implement the operations in an interoperable manner.

The architecture of web service interacts among three roles: service provider, service requester, and service registry. The interaction involves the three operations: publish, find, and bind. These operations and roles act upon the web services artifacts. The web service artifacts are the web service software module and its description.

The service provider hosts a network-associable module (web service). It defines a service description for the web service and publishes it to a service requestor or service registry. These service requestor uses a find operation to retrieve the service description locally or from the service registry. It uses the service description to bind with the service provider and invoke with the web service implementation.




Web Service Roles, Operations and Artifacts

# Practical No: 05

**Aim: Show the implementation of web services.**

## What Are Web Services?

Web services are client and server applications that communicate over the World Wide Web's (WWW) Hypertext Transfer Protocol (HTTP). As described by the World Wide Web Consortium(W3C), web services provide a standard means of interoperating between software applications running on a variety of platforms and frameworks. Web services are characterized by their great interoperability and extensibility, as well as their machine-processable descriptions, thanks to the use of XML. Web services can be combined in a loosely coupled way to achieve complex operations. Programs providing simple services can interact with each other to deliver sophisticated added-value services.

## Types of Web Services:

On the conceptual level, a service is a software component provided through a network- accessible endpoint. The service consumer and provider use messages to exchange invocation request and response information in the form of self-containing documents that make very few assumptions about the technological capabilities of the receiver.

On a technical level, web services can be implemented in various ways. The two types of web services can be distinguished as "big" web services and "RESTful" web services.

### 1) "Big" WebServices:

In Java EE 6, JAX-WS provides the functionality for "big" web services. Big web services use XML messages that follow the Simple Object Access Protocol (SOAP) standard, an XML language defining a message architecture and message formats. Such systems often contain a machine-readable description of the operations offered by the service, written in the Web Services Description Language (WSDL), an XML language for defining interfaces syntactically.

The SOAP message format and the WSDL interface definition language have gained widespread adoption. Many development tools, such as NetBeans IDE, can reduce the complexity of developing web service applications.

A SOAP-based design must include the following elements.

A formal contract must be established to describe the interface that the web service offers. WSDL can be used to describe the details of the contract, which may include messages, operations, bindings, and the location of the web service. You may also process SOAP messages in a JAX-WS service without publishing a WSDL.

The architecture must address complex non-functional requirements. Many web service specifications address such requirements and establish a common vocabulary for them. Examples include transactions, security, addressing, trust, coordination, and soon.

The architecture needs to handle asynchronous processing and invocation. In such cases, the infrastructure provided by standards, such as Web Services Reliable Messaging

(WSRM), and APIs, such as JAX-WS, with their client-side asynchronous invocation support, can be leveraged out of the box.

### 2) RESTful Web Services:

In Java EE 6, JAX-RS provides the functionality for Representational State Transfer (RESTful) web services. REST is well suited for basic, ad hoc integration scenarios. RESTful web services, often better integrated with HTTP than SOAP-based services are, do not require XML messages or WSDL service–API definitions.

Project Jersey is the production-ready reference implementation for the JAX-RS specification. Jersey implements support for the annotations defined in the JAX-RS specification, making it easy for developers to build RESTful web services with Java and the Java Virtual Machine (JVM).

Because RESTful web services use existing well-known W3C and Internet Engineering Task Force (IETF) standards (HTTP, XML, URI, MIME) and have a lightweight infrastructure that allows services to be built with minimal tooling, developing RESTful web services is inexpensive and thus has a very low barrier for adoption. You can use a development tool such as NetBeans IDE to further reduce the complexity of developing RESTful web services.

Web service delivery or aggregation into existing web sites can be enabled easily with a RESTful style. Developers can use such technologies as JAX-RS and Asynchronous JavaScript with XML (AJAX) and such toolkits as Direct Web Remoting (DWR) to consume the services in their web applications. Rather than starting from scratch, services can be exposed with XML and consumed by HTML pages without significantly refactoring the existing web site architecture. Existing developers will be more productive because they are adding to something they are already familiar with rather than having to start from scratch with new technology.

A RESTful design may be appropriate when the following conditions are met.

The web services are completely stateless. A good test is to consider whether the interaction can survive a restart of the server.

A caching infrastructure can be leveraged for performance. If the data that the web service returns is not dynamically generated and can be cached, the caching infrastructure that web servers and other intermediaries inherently provide can be leveraged to improve performance. However, the developer must take care because such caches are limited to the HTTP GET method for most servers.

## Deciding Which Type of Web Service to Use:

Basically, you would want to use RESTful web services for integration over the web and use big web services in enterprise application integration scenarios that have advanced quality of service (QoS) requirements.

**JAX-WS:** addresses advanced QoS requirements commonly occurring in enterprise computing. When compared to JAX-RS, JAX-WS makes it easier to support the WS-* set of protocols, which provide standards for security and reliability, among other things, and interoperate with other WS-* conforming clients and servers.

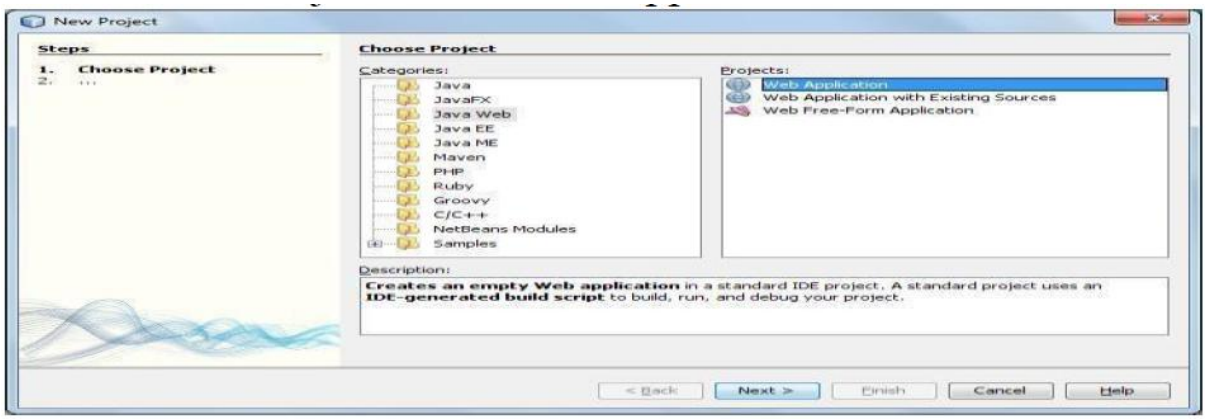
**JAX-RS:** makes it easier to write web applications that apply some or all of the constraints of the REST style to induce desirable properties in the application, such as loose coupling (evolving the server is easier without breaking existing clients), scalability (start small and grow), and architectural simplicity (use off-the-shelf components, such as proxies or HTTP routers). You would choose to use JAX-RS for your web application because it is easier for many types of clients to consume RESTful web services while enabling the server side to evolve and scale. Clients can choose to consume some or all aspects of the service and mash it up with other web-based services.

# Practical 5: Implementing "Big" Web Service.

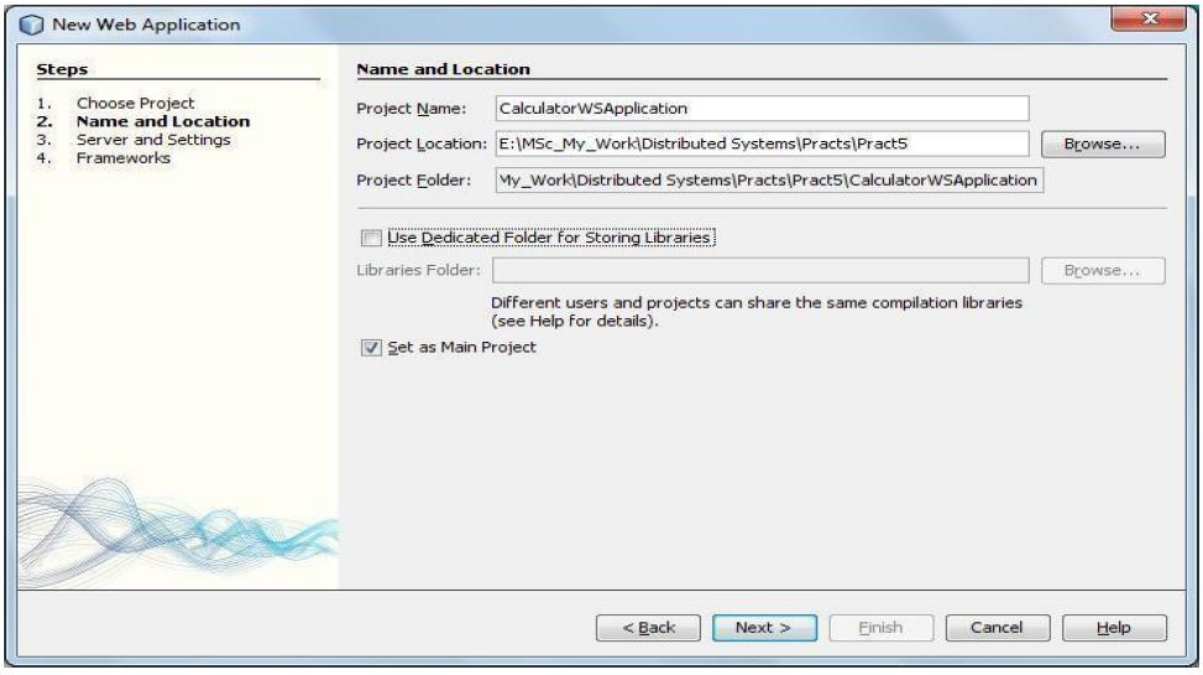
## 1) Creating a Web Service

### A. Choosing a Container:

1. Choose File > New Project. Select Web Application from the Java Web.



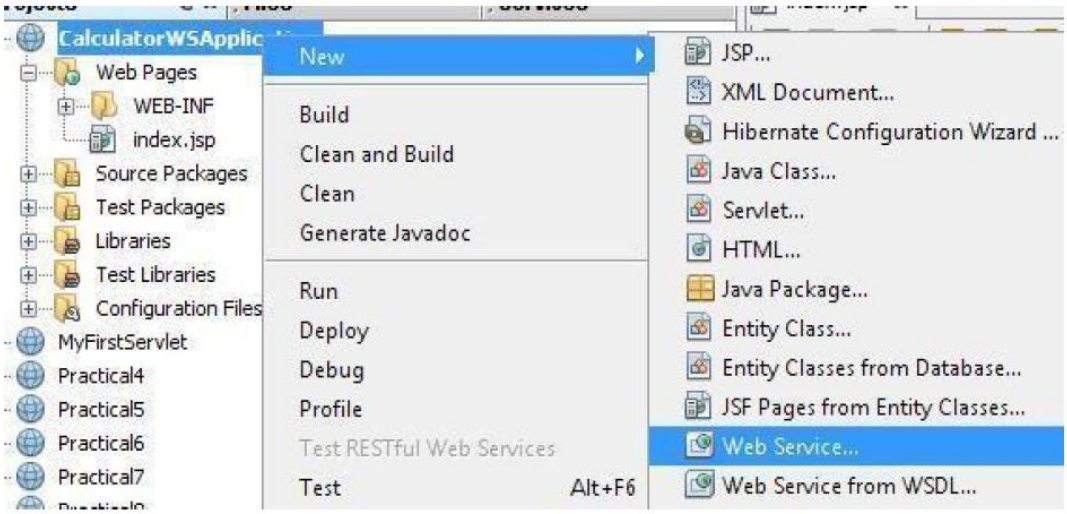

2. Name the project CalculatorWSApplication. Select a location for the project. Click Next.



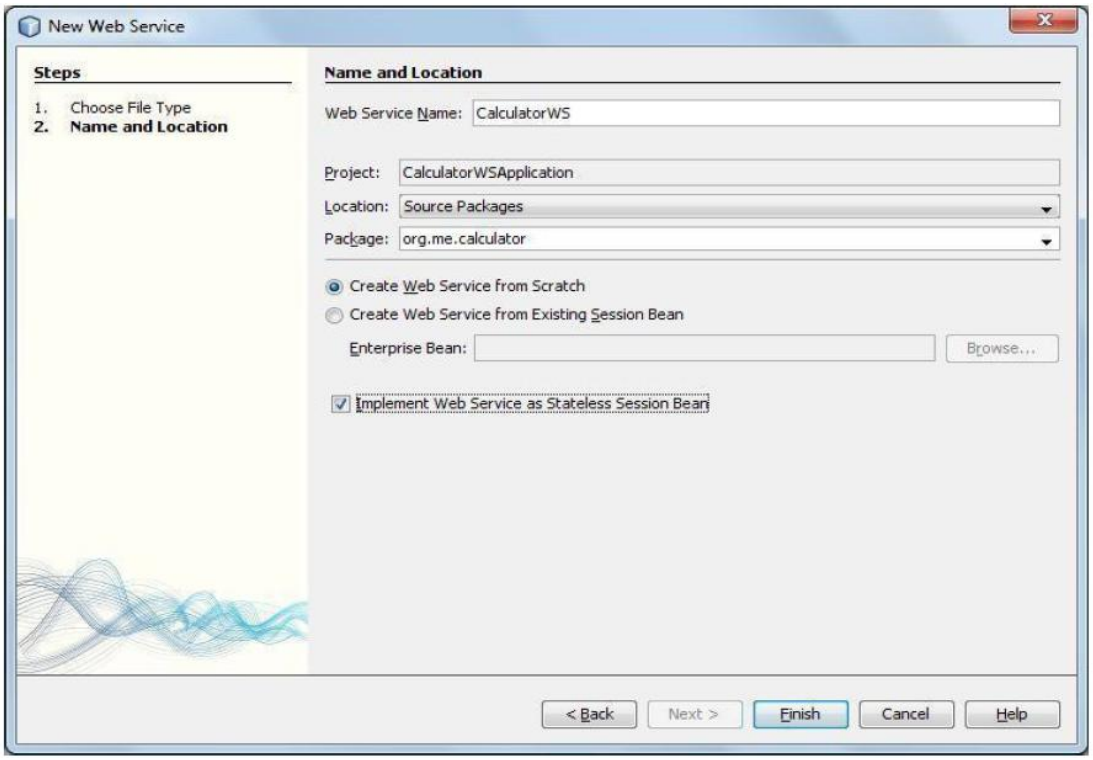

3. Select your server and Java EE version and click Finish.

**B. Creating a Web Service from a Java Class**

1. Right-click the CalculatorWSApplication node and choose New > Web Service.

2. 


Name the web service CalculatorWS and type org.me.calculator in Package. Leave Create Web Service from Scratch selected. If you are creating a Java EE 6 project on GlassFish or WebLogic, select Implement Web Service as a Stateless Session Bean.

3. 


Click Finish. The Projects window displays the structure of the new web service and the source code is shown in the editor area.

**2) Adding an Operation to the Web Service**

The goal of this exercise is to add to the web service an operation that adds two numbers

received from a client. The NetBeans IDE provides a dialog for adding an operation to a web service. You can open this dialog either in the web service visual designer or in the web service
context menu.

**A. To add an operation to the web service:**

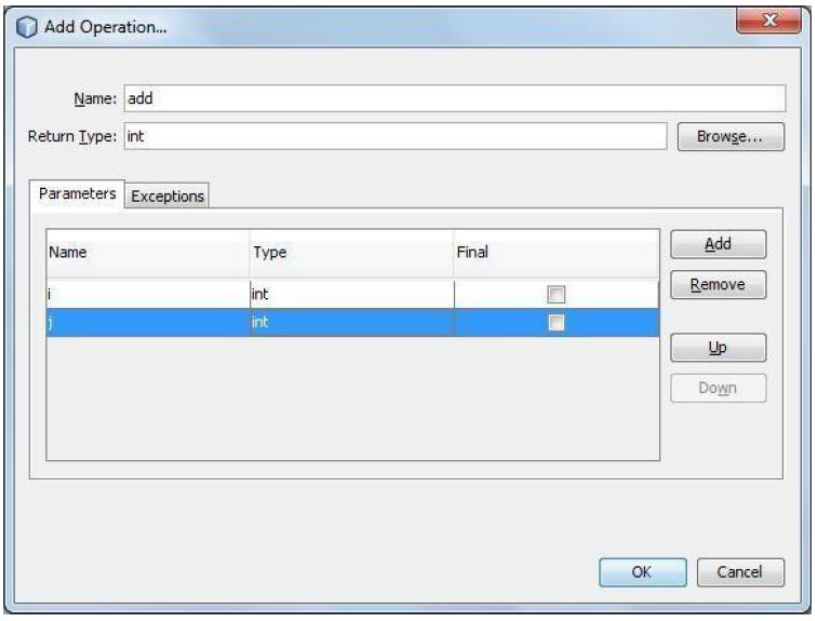
1. Change to the Design view in the editor.


2. Click Add Operation in either the visual designer or the context menu. The Add Operation dialog opens.

3. In the upper part of the Add Operation dialog box, type add in Name and type int in the Return Type drop-down list.

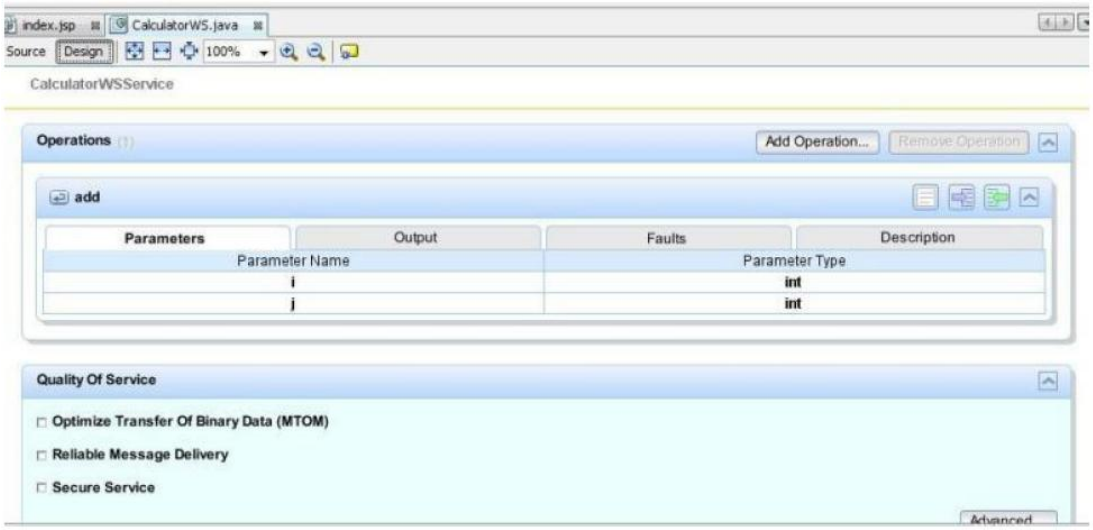
4. In the lower part of the Add Operation dialog box, click Add and create a parameter of type int named i.

5. Click Add again and create a parameter of type int called j. You now see the following:




6. Click OK at the bottom of the Add Operation dialog box. You return to the editor.

7. The visual designer now displays the following:

8. 


Click Source. And code the following.

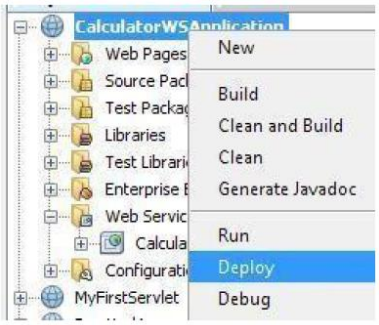
```
@WebMethod(operationName = "add")
public int add(@WebParam(name = "i") int i, @WebParam(name = "j") int j)
{
   int k = i + j;
   return k;
}
```

**3) Deploying and Testing the Web Service**

After you deploy a web service to a server, you can use the IDE to open the server's test client, if the server has a test client. The GlassFish and WebLogic servers provide test clients.

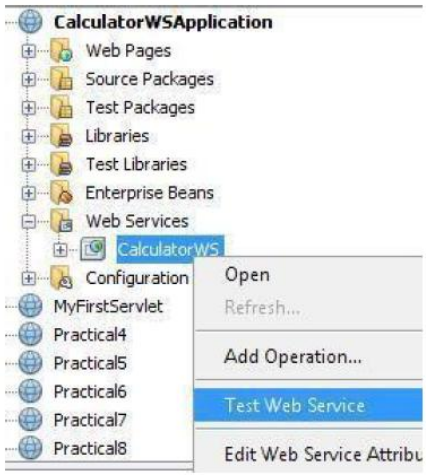
**A. To test successful deployment to a GlassFish or WebLogic server:**

1. Right-click the project and choose Deploy. The IDE starts the application server, builds the application, and deploys the application to the server.




2. In the IDE's Projects tab, expand the Web Services node of the CalculatorWSApplication project. Right-click the CalculatorWS node, and choose Test

Web Service.




3. The IDE opens the tester page in your browser, if you deployed a web application to the GlassFish server.

4. If you deployed to the GlassFish server, type two numbers in the tester page, as shown below:

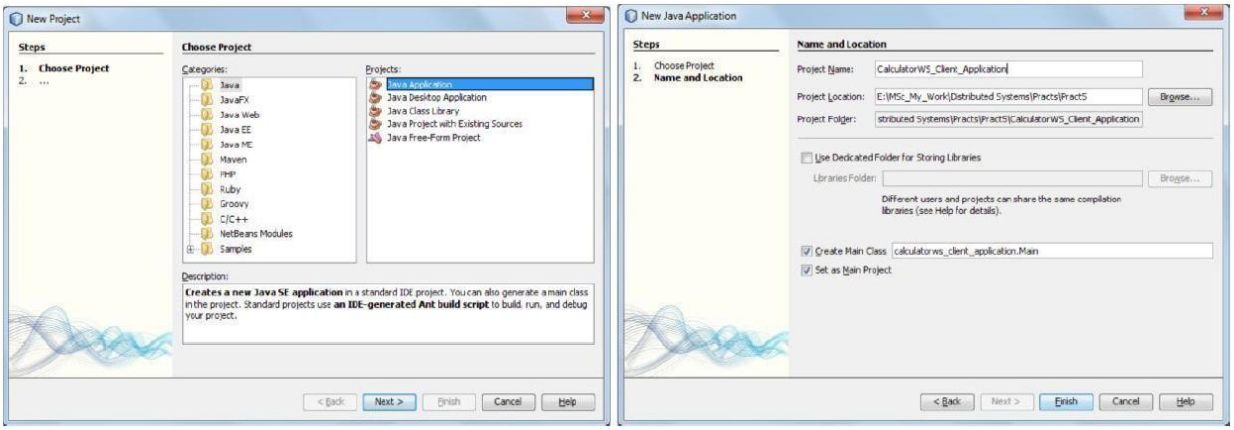

4. The sum of the two numbers is displayed:

**4) Consuming the Web Service**

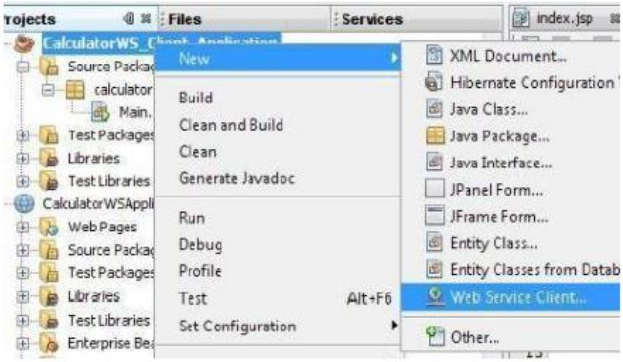
Now that you have deployed the web service, you need to create a client to make use of the web service's add method.

1. **Client: Java Class in Java SE Application**

1. Choose File > New Project. Select Java Application from the Java category. Name the project CalculatorWS_Client_Application. Leave Create Main Class selected and accept all other default settings. Click Finish.

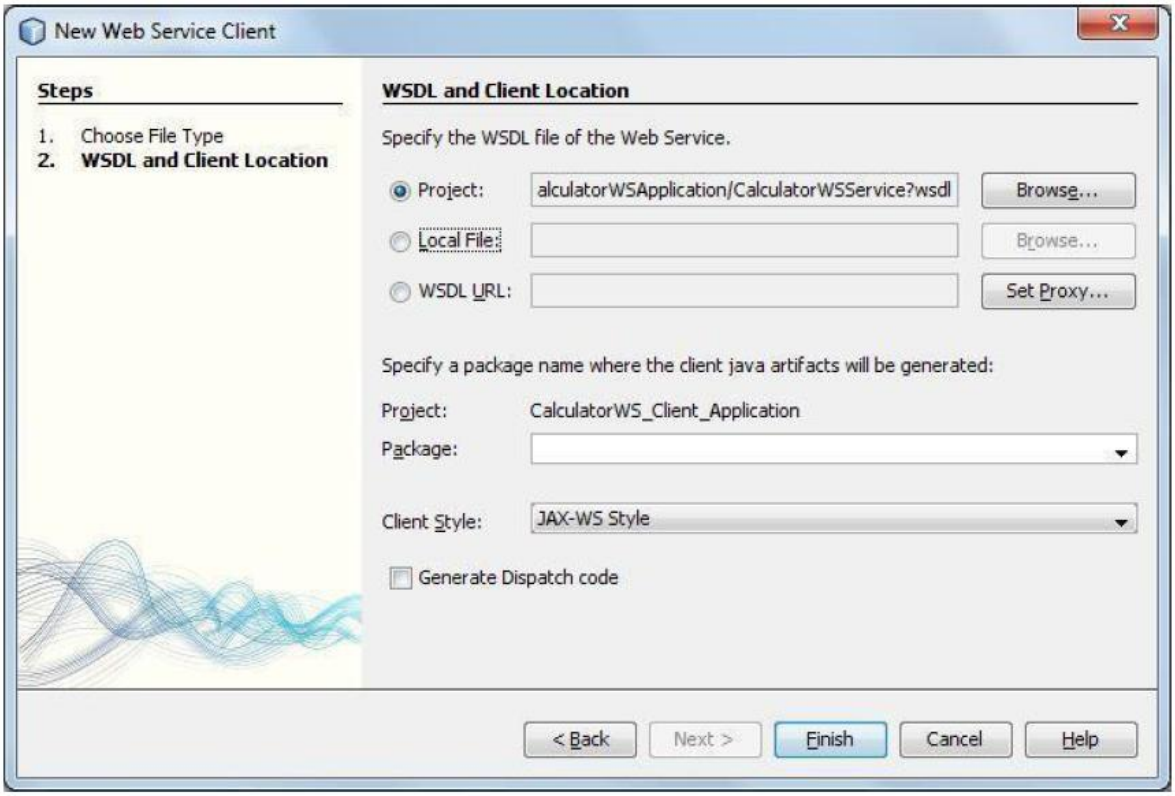



2. Right-click the CalculatorWS_Client_Application node and choose New > Web Service Client. The New Web Service Client wizard opens.



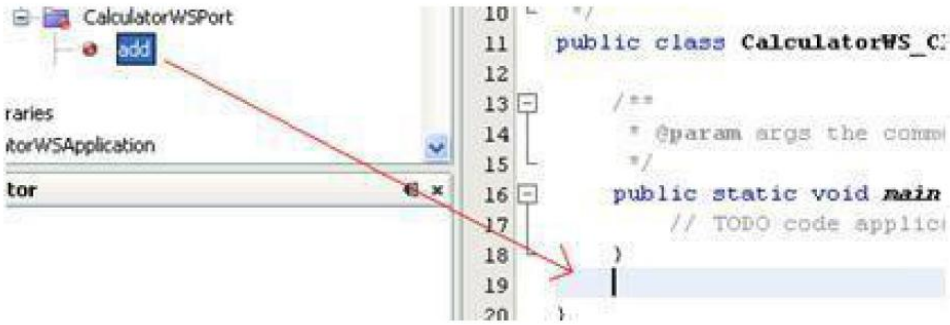

3. Select Project as the WSDL source. Click Browse. Browse to the CalculatorWS web service in the CalculatorWSApplication project. When you have selected the web service, click OK.

4. Do not select a package name. Leave this field empty.




5. Leave the other settings at default and click Finish. The Projects window displays the new web service client, with a node for the add method that you created:

6. Double-click your main class so that it opens in the Source Editor. Drag the add node below the main() method.




You now see the following:

```
public static void main(String[] args)
{
   // TODO code application logic here
}
private static int add(int i, int j)
{
   org.me.calculator.CalculatorWS_Service service = new
   org.me.calculator.CalculatorWS_Service();
   org.me.calculator.CalculatorWS port =
   service.getCalculatorWSPort(); return port.add(i, j);
}
```

7. In the main() method body, replace the TODO comment with code that initializes values for i and j, calls add(), and prints the result.

```
public static void main(String[] args)
{
   int i = 3;
   int j = 4;
   int result = add(i, j);
   System.out.println("Result = " + result);
}
```

8. Surround the main() method code with a try/catch block that prints an exception.

```
public static void main(String[] args)
{
  try
    {
       int i = 3;
       int j = 4;
       int result = add(i, j);
       System.out.println("Result = " + result);
    }
 catch (Exception ex)
   {
    System.out.println("Exception: " + ex);
   }
}
```

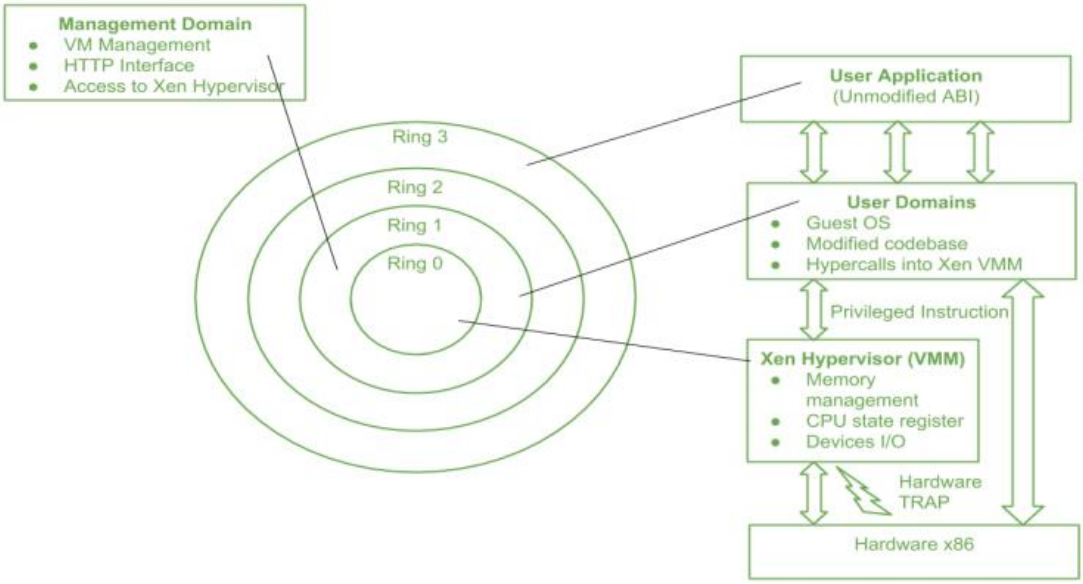
9. Right-click the project node and choose Run.

The Output window now shows the sum:
compile:
run:

Result = 7
BUILD SUCCESSFUL (total time: 1 second)

## Practical No.6

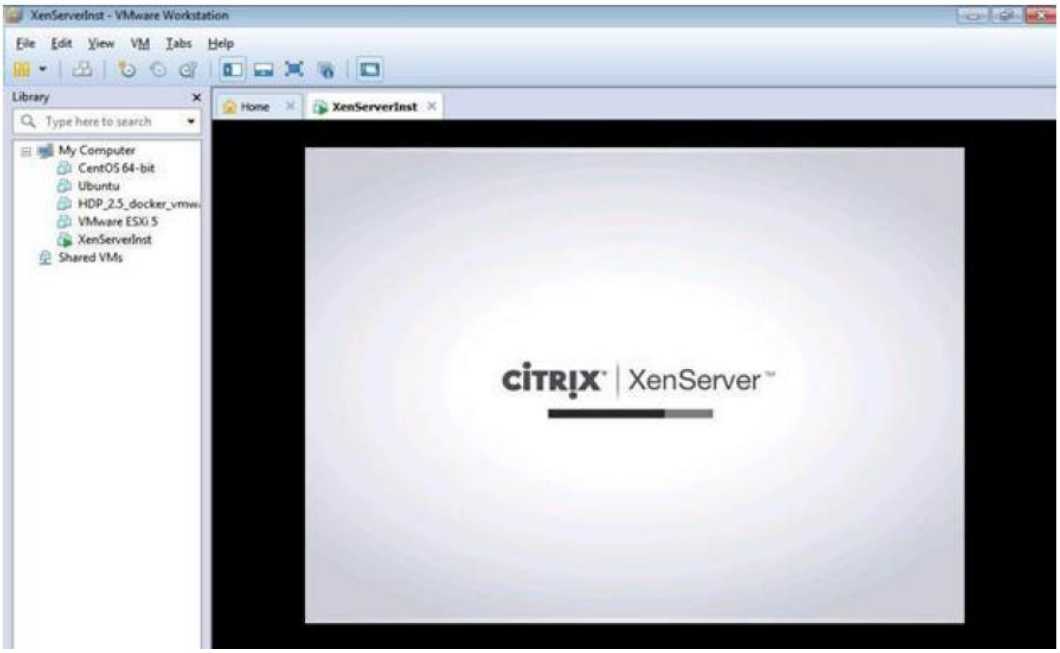
**Xen is an open source hypervisor based on paravirtualization. It is the most popular application of paravirtualization. Xen has been extended to compatible with full virtualization using hardware-assisted virtualization. It enables high performance to execute guest operating system. This is probably done by removing the performance loss while executing the instructions requiring significant handling and by modifying portion of the guest operating system executed by Xen, with reference to the execution of such instructions. Hence this especially support x86, which is the most used architecture on commodity machines and servers.**



# Practical: 06

## Aim: Implement Xen virtualization and manage with Xen Center

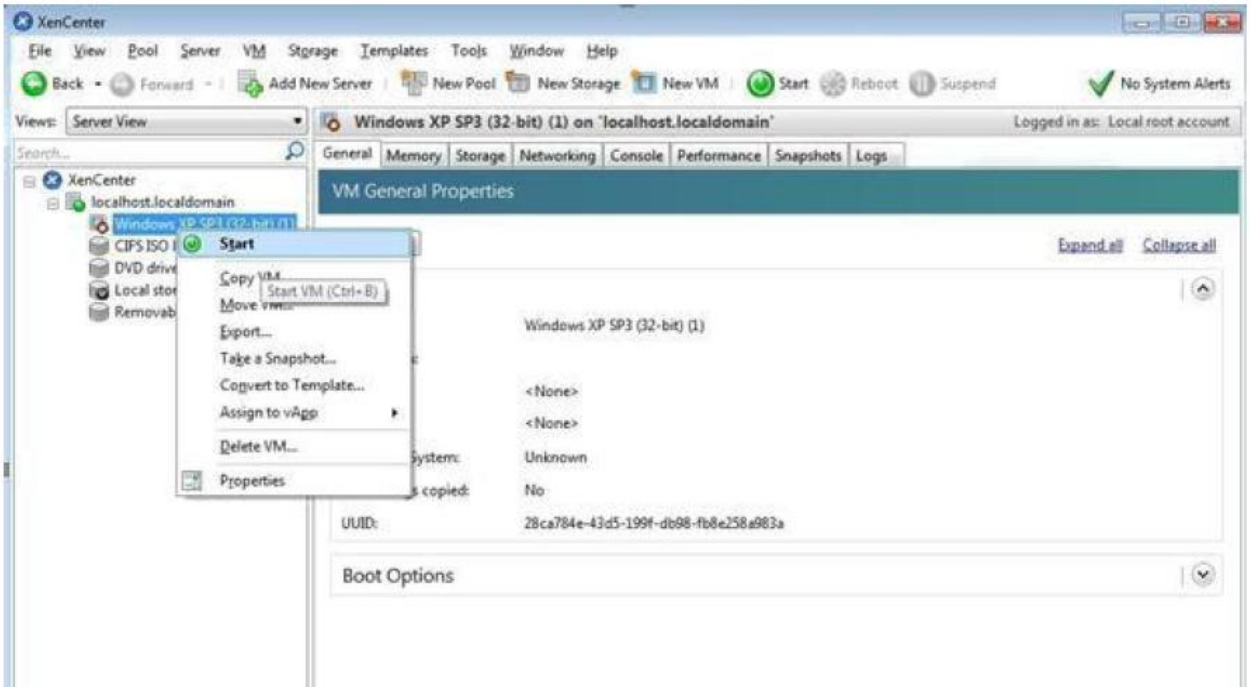
- Install **XenServer** in VMware Workstation and select Guest operating system as Linux.



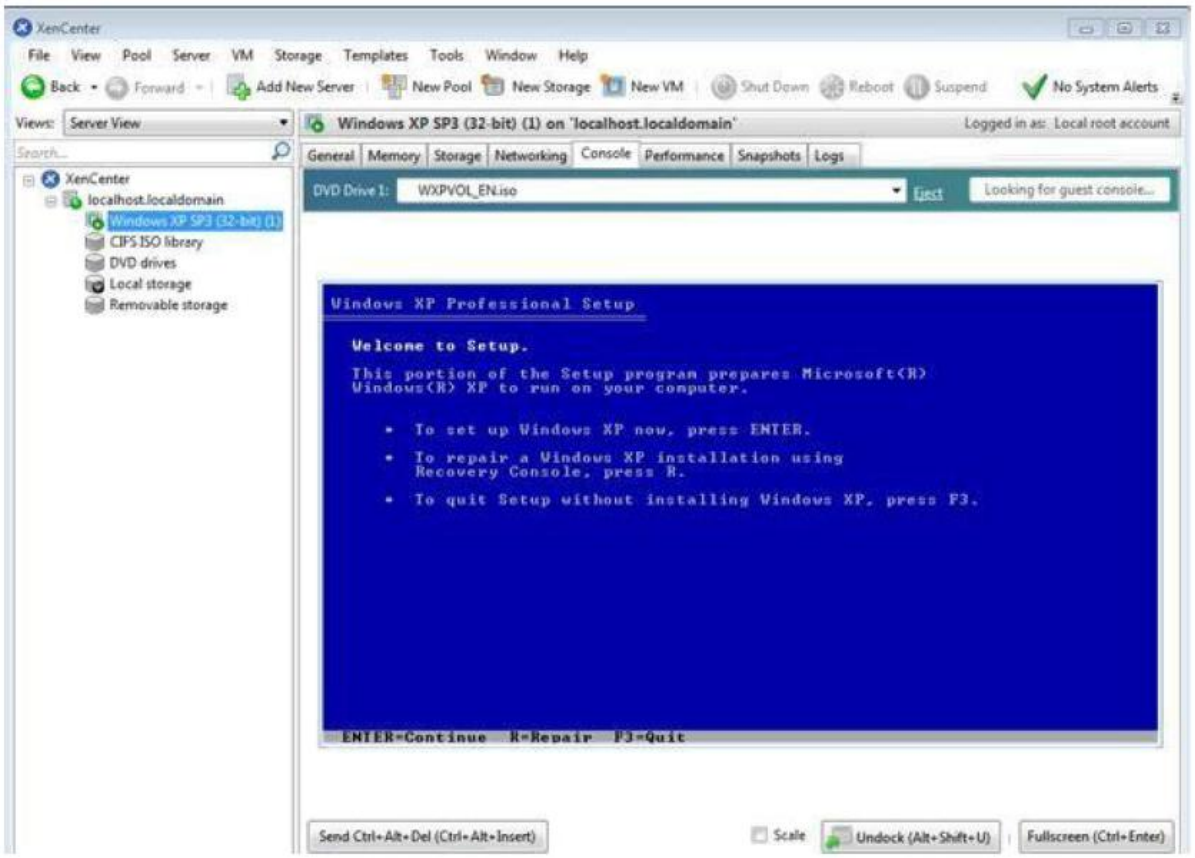

***Note IP Address – " 192.168.124.137" ping it from command prompt.***

- Now Install Citrix App if not installed
- Now Open Citrix XenCenter – and Click and **Add Server.**

- Fill IP address copied from Installation and User name as “root” and Password as “root123” which we had given during installation and Click on Add.
- Then click on Ok
- Now Click on New Storage
- Select Window File Sharing (CIFS) and click on next
- Uncheck Auto generate option Click on Next.
- Provide the path of shared windows XP image and enter local pc credential, click on Finish
- Click on New VM – and Windows XP SP3
- Select ISO file and click on next –
- Click on Next –
- Uncheck – Start the new VM and click on create now
- Now Right click on Windows XP and Start -

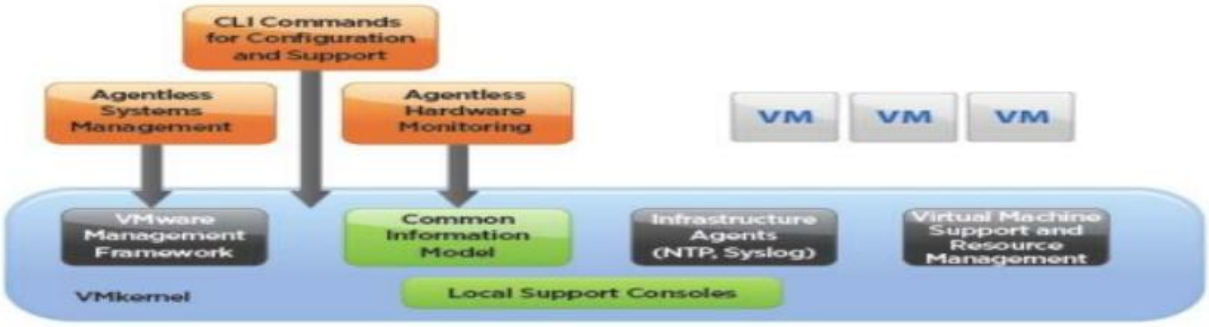


Installation is successful and virtual node has been created if we get below Welcome screen of Windows XP machine.



## Practical No.7

**VMware ESXi is the next-generation hypervisor, providing a new foundation for virtual infrastructure. This innovative architecture operates independently from any general purpose operating system, offering improved security, increased reliability, and simplified management. This paper describes the architecture and operation of VMware ESXi and discusses the new management model associated with it.**

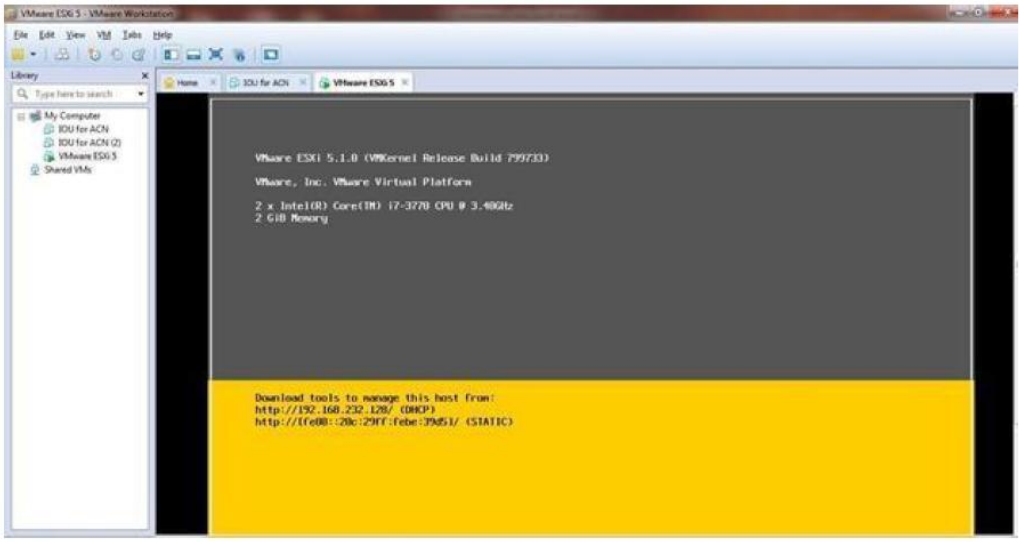


# Practical: 07

## Aim: Implement virtualization using VMWare ESXi Server and managing with vCenter

**Steps:**

Install **ESXi iso** in VMWare workstation.



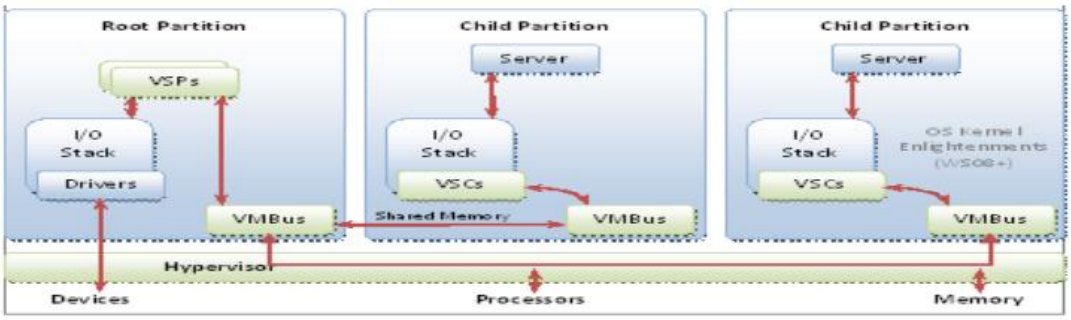

**Install VMware vSphere Client**

In vSphere create new **Virtual Machine.** Install Windows XP iso file and open it.

## Practical No.8

**Hyper-V is a hypervisor-based virtualization technology for certain x64 versions of Windows. The hypervisor is core to virtualization. It is the processor-specific virtualization platform that allows multiple isolated operating systems to share a single hardware platform.**

**Hyper-V supports isolation in terms of a partition. A partition is a logical unit of isolation, supported by the hypervisor, in which operating systems execute. The Microsoft hypervisor must have at least one parent, or root, partition, running Windows. The virtualization management stack runs in the parent partition and has direct access to hardware devices. The root partition then creates the child partitions which host the guest operating systems. A root partition creates child partitions using the hypercall application programming interface (API).**



# Practical: 08

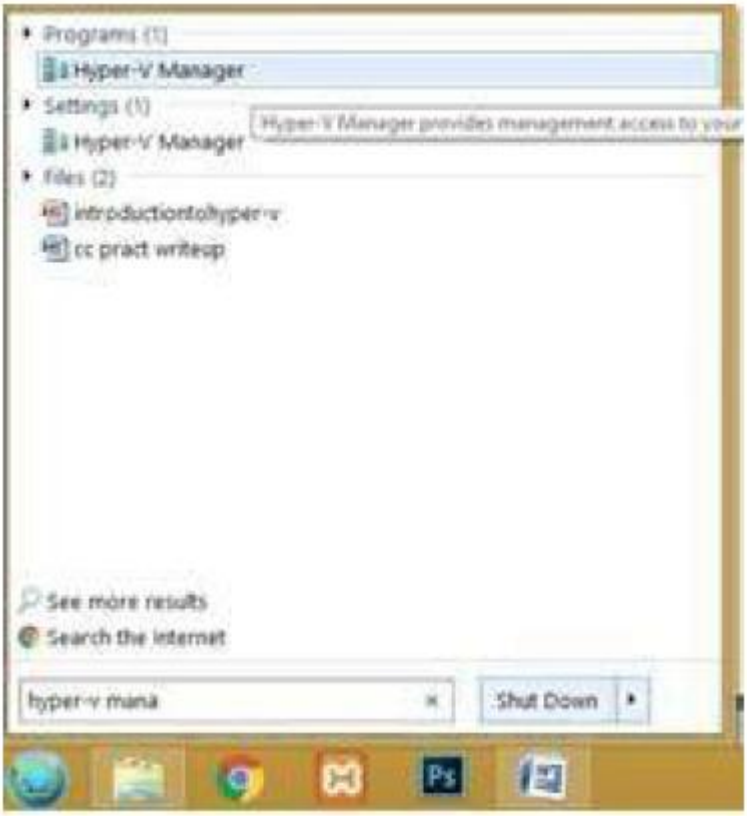
## Aim: Implement Windows Hyper V virtualization

First we have to uninstall vmware software if already installed on computer because the VMware Workstation installer does not support running on a Hyper-V virtual machine. after uninstalling vmware we can proceed to next step go to control panel and click on uninstall a program.

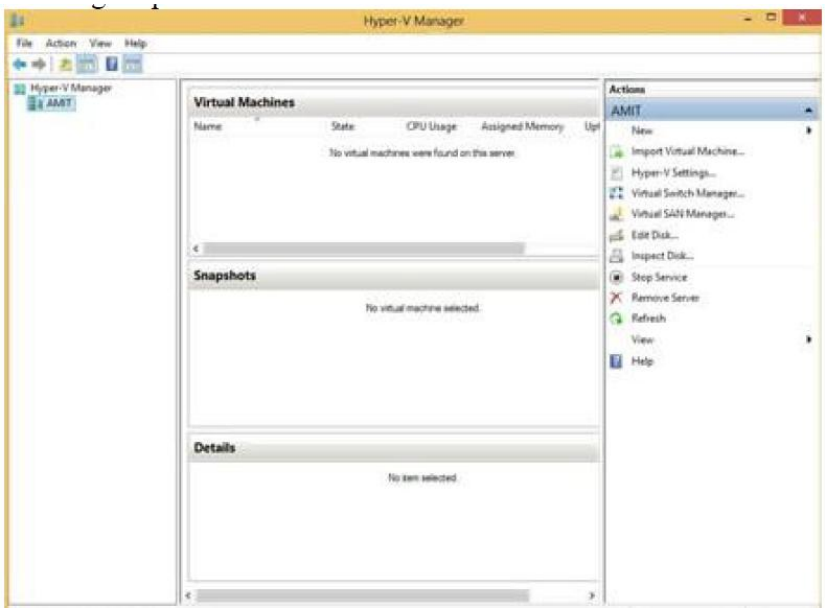

Click on Turn windows features on or off.

Now in windows features check on Hyper-V option.


After Restart **Search for hyper-v manager in search box and click on that.**

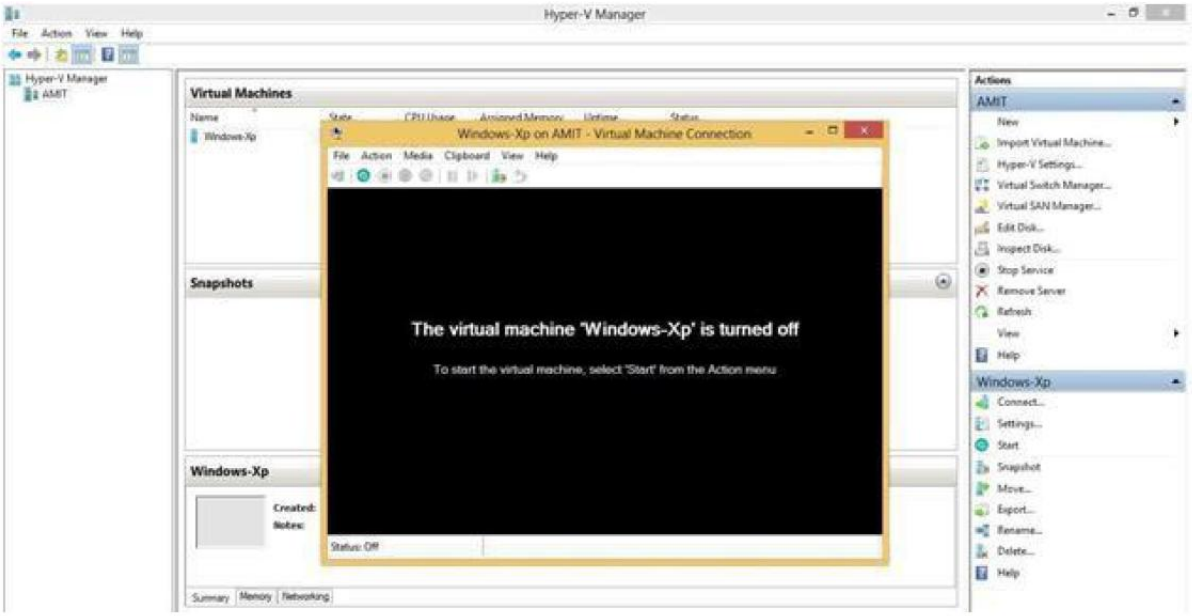



**for creating virtual machine first we have to create virtual switch** click on virtual switch manager option.




**for creating virtual machine first we have to create virtual switch** click on virtual switch manager option.

Select External as a connection type and then click on create virtual switch.
Create new Virtual switch and install windows XP .iso




**And virtual machine will start.**


Welcome to Setup.
This portion of the Setup program prepares Microsoft(R)
Windows(R) XP to run on your computer.
• To set up Windows XP now, press ENTER.
• To repair a Windows XP installation using
Recovery Console, press R.
• To quit Setup without installing Windows XP, press F3.
ENTER=Continue R=Repair F3=Quit