|  |  |  |
| --- | --- | --- |
| **Ex. No** | **LIST OF EXPERIMENT** | **Page No.** |
| 1 | **STUDY EXPERIMENTS** |  |
|  | * 1. STUDY OF BASICS OF NETWORKING |  |
|  | 1.2 STUDY OF COMMANDS FOR SOCKET PROGRAMMING |  |
| **2** | **BASIC PROGRAMS** |  |
|  | 2.1 GENERATE RANDOM PORT NUMBER |  |
|  | 2.2 FIND IP ADDRESS |  |
| **3** | **REMOTE COMMAND EXECUTION** |  |
| **4** | **TCP COMMUNICATION** |  |
|  | 4.1 ONE WAY COMMUNICATION |  |
|  | 4.2 TWO WAY COMMUNICATION |  |
|  | 4.3 ECHO COMMAND |  |
|  | 4.4 PING COMMAND |  |
|  | 4.5 TALK COMMAND |  |
|  | 4.6 FILE TRANSFER USING TCP |  |
| **5** | **UDP COMMUNICATION** |  |
|  | 5.1 ONE WAY COMMUNICATION |  |
|  | 5.2 TWO WAY COMMUNICATION |  |
| **6** | **SIMPLE REMOTE METHOD INVOCATION** |  |
| **7** | **REMOTE METHOD INVOCATIONAPPLICATION** |  |
| **8** | **ADDRESS RESOLUTION PROTOCOL** |  |
| **9** | **CYCLIC REDUNDANCY CHECK** |  |
| **10** | **SLIDING WINDOW PROTOCOL** |  |
| **11** | **HTTP WEBPAGE DOWNLOAD(RESPONSE CODE)** |  |
| **12** | **ROUTING** |  |
|  | 11.1 SHORTEST PATH ROUTING |  |
|  | 11.2MULTICAST ROUTING |  |
| **13** | **IMPLEMENTATION USING NS2** |  |
| **14** | **ASSIGNMENTS** |  |
|  | **14.1 STOP AND WAIT PROTOCOL** |  |
|  | **14.2 Implementation of Sub netting** |  |
|  | **14.3 TRACEROUTING** |  |
|  | **14.4 DNS** |  |
|  | **14.5 SNMP** |  |
| **15** | **VIVA Questions** |  |

|  |  |
| --- | --- |
| **Ex.No:1.a** | **STUDY OF BASICS OF NETWORKING** |
| **Date:** |

# Aim:

To study the basics in networking

**a)Network Device**

Study of following Network Devices in Detail

• Repeater

• Hub

• Switch

• Bridge

• Router

• Gate Way

* Protocol

1. **Repeater:** Functioning at Physical Layer. A **repeater** is an electronic device that receives a

signal and retransmits it at a higher level and/or higher power, or onto the other side of an

obstruction, so that the signal can cover longer distances. Repeater have two ports ,so cannot be

use to connect for more than two devices

2. **Hub:** An **Ethernet hub**, **active hub**, **network hub**, **repeater hub**, **hub** or **concentrator**

is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and

making them act as a single network segment. Hubs work at the physical layer (layer 1) of the

OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision

detection, forwarding a jam signal to all ports if it detects a collision.

3. **Switch:** A **network switch** or **switching hub** is a computer networking device that connects

network segments. The term commonly refers to a network bridge that processes and routes data

at the data link layer (layer 2) of the OSI model. Switches that additionally process data at the

network layer (layer 3 and above) are often referred to as Layer 3 switches or multilayer

switches.

4. **Bridge:** A **network bridge** connects multiple network segments at the data link layer (Layer

2) of the OSI model. In Ethernet networks, the term *bridge* formally means a device that behaves

according to the IEEE 802.1D standard. A bridge and switch are very much alike; a switch being

a bridge with numerous ports. *Switch* or *Layer 2 switch* is often used interchangeably with

*bridge* .Bridges can analyze incoming data packets to determine if the bridge is able to send the

given packet to another segment of the network.

5. **Router:** A **router** is an electronic device that interconnects two or more computer networks,

and selectively interchanges packets of data between them. Each data packet contains address

information that a router can use to determine if the source and destination are on the same

network, or if the data packet must be transferred from one network to another. Where multiple

routers are used in a large collection of interconnected networks, the routers exchange

information about target system addresses, so that each router can build up a table showing the

preferred paths between any two systems on the interconnected networks.

6. **Gate Way:** In a communications network, a network node equipped for interfacing with

another network that uses different protocols.

• A gateway may contain devices such as protocol translators, impedance matching

devices, rate converters, fault isolators, or signal translators as necessary to provide

system interoperability. It also requires the establishment of mutually acceptable

administrative procedures between both networks.

**7.Protocol**

A protocol translation/mapping gateway interconnects networks with different network

protocol technologies by performing the required protocol conversions.

**b)Study of network IP , Ports**

An Internet Protocol address (IP address) is a numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication. An IP address serves two principal functions: host or network interface identification and location addressing.

**Classification of IP address**

**Sub netting**

The subnet mask is used by the TCP/IP protocol to determine whether a host is on the local subnet or on a remote network. So now you know, for this example using a **255.255.255.0** subnet mask, that the network ID is **192.168.123.0**, and the host address is 0.0.0.132

**Super netting**

A super network, or super net, is an [Internet Protocol](https://en.wikipedia.org/wiki/Internet_Protocol) (IP) network that is formed, for routing purposes, from the combination of two or more networks (or [subnets](https://en.wikipedia.org/wiki/Subnetwork)) into a larger network. The new [routing prefix](https://en.wikipedia.org/wiki/Routing_prefix) for the combined network represents the constituent networks in a single [routing table](https://en.wikipedia.org/wiki/Routing_table) entry. The process of forming a super net is called super netting, prefix aggregation, route aggregation, or route summarization.

**Network Masks**

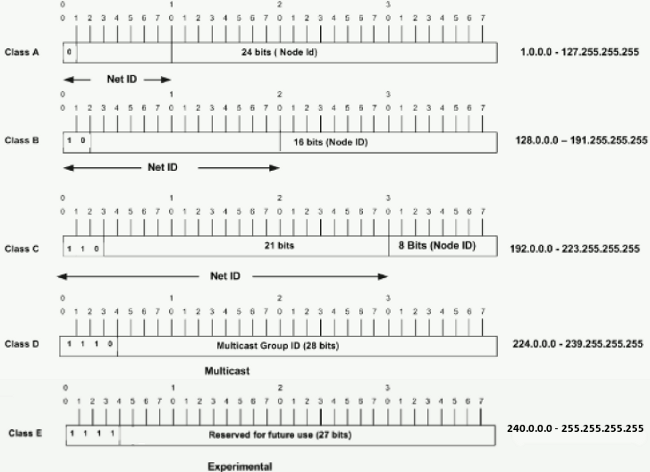
A network mask helps you know which portion of the address identifies the network and which portion of the address identifies the node. Class A, B, and C networks have default masks, also known as natural masks, as shown here:

Class A: 255.0.0.0

Class B: 255.255.0.0

Class C: 255.255.255.0

|  |  |  |
| --- | --- | --- |
| **Class** | **Address Range** | **Supports** |
| **Class A** | 1.0.0.0 to 127. 255.255.254 | Supports 16 million hosts on each of 127 networks |
| **Class B** | 128.0.0.0 to 191.255.255.254 | Supports 65,000 hosts on each of 16,000 networks. |
| **Class C** | 192.0.0.0 to 223. 255.254.254 | Supports 254 hosts on each of 2 million networks. |
| **Class D** | 224.0.0.0 to 239.255.255.255 | Reserved for multicast groups |
| **Class E** | 240.0.0.0 to 254.255.255.254 | Reserved. |



**Ports**

An IP address identifies a host machine on the Internet. An IP port will identify a specific application running on an Internet host machine. A port is identified by a number, the port number. -The number of ports is not functionally limited, in contrast to serial communications where only 4 ports are allowed.

Number of ports: 65,536  
Range: 0 – 65535

**Types of Ports**

|  |  |
| --- | --- |
| Port Type | Port no. |
| Well Known | 0 – 1023 |
| Registered Ports | 1024 – 49151 |
| Ephemeral Ports | 49152 – 65535 |

Type “***netstat -a***” in the command prompt and press ‘Enter’, this lists all the ports being used

**Socket:**  
The unique combination of IP address and Port number together are termed as Socket.

**List of Well-Known Ports**

Port numbers range from 0 to 65535, but only port numbers 0 to 1023 are reserved for privileged services and designated as well-known ports. The following list of *well-known port numbers* specifies the port used by the server process as its contact port.

|  |  |
| --- | --- |
| **Port Number ---** | **Description** |
| 1 | [TCP](https://www.webopedia.com/TERM/T/TCP.html) Port Service Multiplexer (TCPMUX) |
| 5 | Remote Job Entry (RJE) |
| 7 | ECHO |
| 18 | Message Send Protocol (MSP) |
| 20 | [FTP](https://www.webopedia.com/TERM/F/ftp.html) -- Data |
| 21 | FTP -- Control |
| 22 | [SSH](https://www.webopedia.com/TERM/S/SSH.html) Remote Login Protocol |
| 23 | [Telnet](https://www.webopedia.com/TERM/T/Telnet.html) |
| 25 | [Simple Mail Transfer Protocol](https://www.webopedia.com/TERM/S/SMTP.html) (SMTP) |
| 29 | MSG ICP |
| 37 | Time |
| 42 | Host Name Server (Nameserv) |
| 43 | WhoIs |
| 49 | Login Host Protocol (Login) |
| 53 | [Domain Name System](https://www.webopedia.com/TERM/D/DNS.html) (DNS) |
| 69 | [Trivial File Transfer Protocol](https://www.webopedia.com/TERM/T/TFTP.html) (TFTP) |
| 70 | [Gopher](https://www.webopedia.com/TERM/G/gopher.html) Services |
| 79 | [Finger](https://www.webopedia.com/TERM/F/finger.html) |
| 80 | [HTTP](http://www.webopedia.com/TERM/H/HTTP.html) |

# Result:

Thus the study of basics of networks is done.

|  |  |
| --- | --- |
| **Ex.No:1.b** | **STUDY OF COMMANDS FOR SOCKET PROGRAMMING** |
| **Date:** |

# Aim:

To study the commands of the socket programming.

# Sockets:

A socket is one end-point of a two-way communication link between two programs running on the network. Socket classes are used to represent the connection between a client program and a server program.

The java.net package provides two classes--Socket and ServerSocket--that implement the client side of the connection and the server side of the connection,respectively.

They connect to them on published ports when the ServerSocket created it will register itself with the system as having an internet in client connection. The constructor for the ServerSocket having clients connection it can leave pending before it should be refers connections

# Port:

The TCP and UDP protocols use ports to map incoming data to a particular process running on a computer. Port numbers range from 0 to 65,535 because ports are represented by 16-bit numbers. The port numbers ranging from 0 - 1023 are restricted; they are reserved for use by well-known services such as HTTP and FTP and other system service( called *well-knownports*.)

# TCP/IP Protocol:

TCP provides a point-to-point channel for applications that require reliable communications. The Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), and Telnet are all examples of applications that require a reliable communication channel. The order in which the data is sent and received over the network is critical to the success of these applications. When HTTP is used to read from a URL, the data must be received in the order in which it was sent. Otherwise, you end up with a jumbled HTML file, a corrupt zip file, or some other invalidinformation

# Stream communication

The stream communication protocol, transfer control protocol, TCP is a connection-oriented protocol. In order to do communication over the TCP protocol, a connection must first be established between the pair of sockets. While one of the sockets listens for a connection request (server), the other asks for a connection.Oncetwosocketshavebeenconnected,theycanbeusedtotransmitdatainbothdirections

# Creating TCP Servers

1. Create a ServerSocket attached to a portnumber.

# ServerSocket server = new ServerSocket(port);

1. Wait for connections from clients requesting connections to thatport.

# Socket channel = server.accept();

1. Get input and output streams associated with thesocket.

# out = new PrintWriter (channel.getOutputStream());

**reader = new InputStreamReader (channel.getInputStream()); in = new BufferedReader (reader);**

**String data = in.readLine(); out.println("Hey! ");**

**Creating TCP Clients**

1. Create a Socket object attached to a remote host,port.

# Socket client = new Socket(host, port);

1. Get input and output streams associated with thesocket.

# out = new PrintWriter (client.getOutputStream());

**reader = new InputStreamReader (client.getInputStream()); in = new BufferedReader (reader);**

**out.println("Hi!”);**

**String data = in.readLine();**

**UDP**

The UDP protocol provides for communication that is not guaranteed between two applications on the network. UDP is not connection-based like TCP. Rather, it sends independent packets of data, called *datagrams*, from one application to another. Sending datagrams is much like sending a letter through the postal service: The order of delivery is not important and is not guaranteed, and each message is independent of any other.

# Datagram communication

The datagram communication protocol, user datagram protocol, is a connectionless protocol, meaning that each time you send datagrams, you also need to send the local socket descriptor and the receiving socket’s address.

# Creating UDP Servers

1. Create a DatagramSocket attached to a port. **int port = 1234;**

# DatagramSocket socket = new DatagramSocket(port);

1. Allocate space to hold the incoming packet, and create an instance ofDatagramPacket to hold the incomingdata.

# byte[] buffer = new byte[1024]; DatagramPacket packet =

**new DatagramPacket(buffer, buffer.length);**

1. Block until a packet is received, then extract the information you need from thepacket. **socket.receive(packet);**

# InetAddress remoteHost = packet.getAddress(); int remotePort = packet.getPort();

**byte[] data = packet.getData(); Creating UDP Clients**

1. First allocate space to hold the data we are sending and create an instance of **byte[] buffer = newbyte[1024];**

# int port = 1234;

**InetAddress host =InetAddress.getByName("magelang.com"); DatagramPacket p =new DatagramPacket(buffer, buffer.length,host, port);**

1. Create a DatagramSocket and send the packet using thissocket.

# DatagramSocket socket = new DatagramSocket(); socket.send(p);

**InetAddress localHostname = socket.getLocalAddress(); int localPort = socket.getLocalPort();**

**SERVERSOCKET CONSTRUCTOR AND METHODS**

The java.net.ServerSocket class is used by server applications to obtain a port and listen for client requests

|  |  |
| --- | --- |
| **CONSTRUCTOR** | **DESCRIPTION** |
| ServerSocket(int port) | Create a server socket bound to the specified port. |
| ServerSocket(int port, int backlog) | The backlog parameter specifies how many incoming clients to store in a wait queue. |
| ServerSocket(int port, int backlog, InetAddressaddress) | The inetaddress specifies the local IP address to bind to and used for servers that may have multiple IP addresses, allowing the server to specify which ofits  IP addresses to accept client requests on |
| ServerSocket() | Creates an unbound server socket. When using this constructor, use the bind() method when you areready to bind the serversocket |

|  |  |
| --- | --- |
| **METHODS** | **DESCRIPTION** |
| getLocalPort() | Returns the port that the server socket is listeningon. |
| Socket accept() | Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that thetime-outvaluehasbeensetusingthe  setSoTimeout() method. |
| setSoTimeout(int timeout) | Sets the time-out value for how long the server socket waits for a client during the accept(). |
| bind(SocketAddress host, int backlog) | Binds the socket to the specified server and port in the SocketAddressobject. |

# SOCKET CONSTRUCTOR AND METHODS

The creation of socket is done through the constructor of the class socket. The creation of the socket objects implicates establishes a connection between the client and the server.

The java.net.Socket class represents the socket that both the client and server use to communicate with each other. The client obtains a Socket object by instantiating one, whereas the server obtains a Socket object from the return value of the accept() method.

|  |  |
| --- | --- |
| **CONSTRUCTOR** | **DESCRIPTION** |
| Socket(String host, int port) | Connect to the specified server at the specified port. |
| Socket(InetAddress host, int port) | The host is denoted by an inetaddress object. |

|  |  |
| --- | --- |
| Socket(String host, int port, InetAddress localAddress, int localPort) | Connects to the specified host and port, creating a socket on the local host at the specified address and port. |
| Socket(InetAddress host, int port, InetAddress localAddress, int localPort) | The host is denoted by an inetaddress object instead of aString |
| Socket() | Creates an unconnected socket. Use the connect()  method to connect this socket to a server. |

|  |  |
| --- | --- |
| **METHODS** | **DESCRIPTION** |
| connect(SocketAddress host, int timeout) | Connects the socket to the specified host. This method is needed only when you instantiated the Socket using the no-argument constructor. |
| InetAddress getInetAddress() | Returns the address of the other computer that this socket is connected to. |
| getPort() | Returns the port the socket is bound to on the remote machine. |
| getLocalPort() | Returns the port the socket is bound to on the local machine. |
| close() | Closes the socket, which makes this Socket object no longer capable of connecting again to any server |

# UNICAST

An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.

**The Unspecified Address**-- Also called anylocal or wildcard address. It must never be assigned to any node. It indicates the absence of an address. One example of its use is as the target of bind, which allows a server to accept a client connection on any interface, in case the server host has multiple interfaces.The unspecified address must not be used as the destination address of an IPpacket.

**The Loopback Addresses**-- This is the address assigned to the loopback interface. Anything sent to this IP address loops around and becomes IP input on the local host. This address is often used when testing a client.

# MULTICAST

An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address

|  |  |
| --- | --- |
| **METHODS** | **DESCRIPTION** |
| InetAddress getByAddress(byte[] addr) | Returns an InetAddress object given the raw IP address . |
| InetAddress getByAddress(String host, byte[] addr) | Create an InetAddress based on the provided host name and IPaddress. |
| InetAddress getByName(String host) | Determines the IP address of a host, given the host's name. |
| getHostAddress() | Returns the IP address string in textual presentation. |
| getHostName() | Gets the host name for this IP address. |
| InetAddress InetAddress getLocalHost() | Returns the local host. |

# Result:

Thus the study of the commands for the socket programming is done.

|  |  |
| --- | --- |
| **Ex. No:2.1** | **GENERATE RANDOM PORT NUMBER** |
| **Date:** |

# Aim:

To write a java program for generate random port number

# Algorithm:

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called RandomPort.
4. InsideRandomPort,declareanobjectofclassServerSocket,called“server”,withportnumber0.
5. Display a message saying which port the object “server” runs on by getting the port number using the methodgetLocalPort().
6. Catch and handle any exceptionsthrown.
7. Stop

**RandomPort.java**

import java.net.\*; import java.io.\*; class RandomPort

{

public static void main(String[] args)

{

try

{

}

ServerSocket server = new ServerSocket(0);

System.out.println("This server runs on port " + server.getLocalport( ));

catch (IOException e)

{

System.err.println(e);

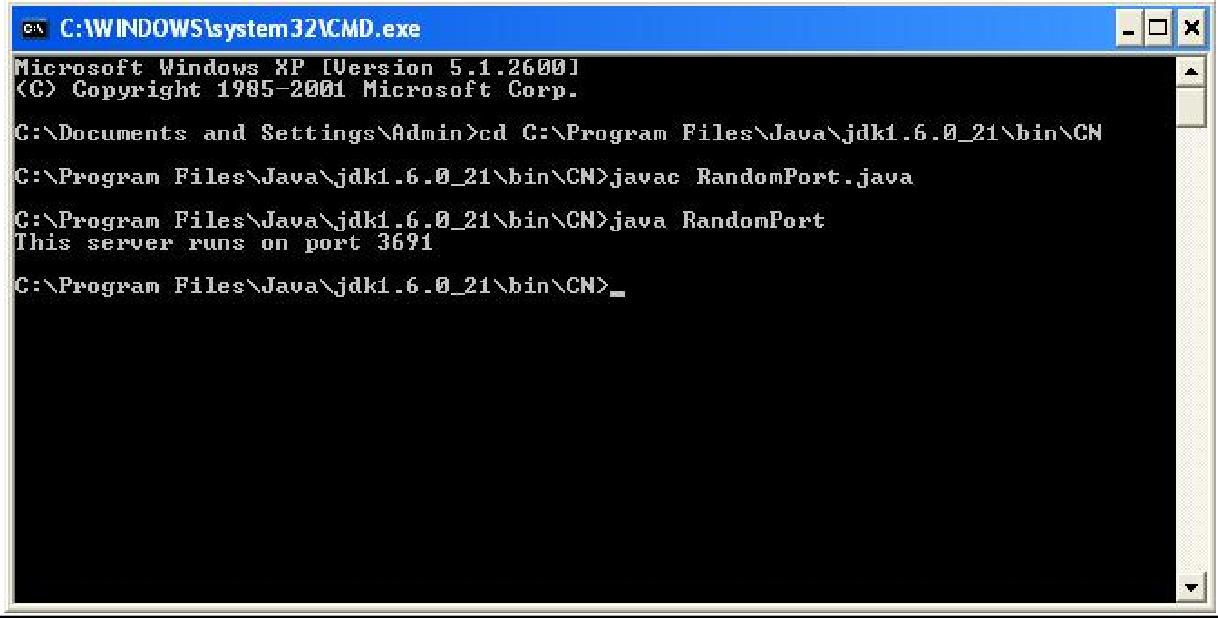
}

}

}

# OUTPUT

**RandomPort.java**



**Result:**

Thus the program random port number generated executed and verified.

|  |  |
| --- | --- |
| **Ex. No:2.2** | **FIND IP ADDRESS OF CLIENT** |
| **Date :** |

# Aim:

To write a java program for Finding an IP address of client.

# Algorithm

1. Start.
2. Import the java.net, java.io, and java.utilpackages.
3. Declare a new class called“GetOwnIp”.
4. Create an object for the InetAddress Method to get the LocalHost IPAddress.
5. Display the message received, which is the local host InetAddress.
6. Stop

# OwnIp.java

import java.util.\*; import java.lang.\*; import java.net.\*; public class GetOwnIP

{

public static void main(String args[])

{

try

{

InetAddress ownIP=InetAddress.getLocalHost(); System.out.println("IP of my system is :="+ownIP.getHostAddress());

}

catch (Exception e)

{

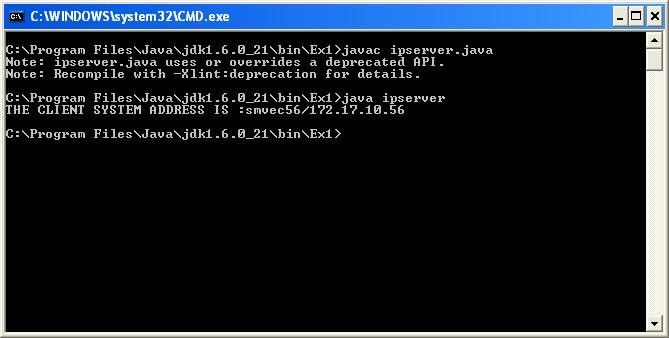
System.out.println("Exception caught ="+e.getMessage());

}

}

}

# Output:



**Result:**

Thus the program finding an IP address of Clinet executed and verified.

|  |  |
| --- | --- |
| **Ex. No:3** | **REMOTE COMMAND EXECUTION** |
| **Date :** |

# Aim:

To write a java program for Remote Command Execution.

# Algorithm Server Program

1. Start.
2. Import the java.net, java.io, and java.langpackages.
3. Declare a new class called “RCEserver”.
4. Within class “RCEserver”, create an object of class ServerSocket called “ss” with the port number1680.
5. Create a Socket object called “cs” using the accept()method.
6. Create an object of class Runtime using the getRuntime()method.
7. Create a new object of class Process called “p” and initialize it tonull;
8. CreateanewDataInputStreamobjectwhichactsasaninputstreamtotheserverfromtheclient.
9. Read in the input from the client using the DataInputStream object and store it in string“s”.
10. Display the value of string“s”.
11. Execute the string command stored in “s” using the Runtime object's exec() method in a separate process“p”.
12. Using the waitFor() method, make the current thread wait until Process “p” has finished executing.
13. Stop.

# Client Program

1. Start.
2. Import the java.net, java.lang, and java.iopackages.
3. Declare a new class called “RCEclient”.
4. Within class “RCEclient”, create a new Socket object called “c” with the port number 1680.
5. Create a new DataInputStream object which acts as an input stream from the user tothe client.
6. Prompt and read in the command from the user using the DataInputStreamobject.
7. Create a new DataOutputStream object which acts as an output stream from the client to the server.
8. Send the command read from the user to the server using the DataOutputStreamobject.
9. Close the input and output streams, and close the Socketobject.
10. Stop.

# RCEserver.java

import java.io.\*; import java.net.\*; import java.lang.\*; class RCEserver

{

public static void main(String a[])throws IOException,UnknownHostException,InterruptedException

{

ServerSocket ss=new ServerSocket(1680); Socket cs=ss.accept();

Runtime r=Runtime.getRuntime();//The current runtime can be obtained from this method

//Allows to interface with the environment in which the application is running. Process p=null;

InputStream is=cs.getInputStream(); DataInputStream dis=new DataInputStream(is); String s=dis.readLine();

System.out.println(s);

p=r.exec(s);//Executes the specified string command in a separate process p.waitFor();//Causes the current thread to wait

}

}

# RCEclient.java

import java.io.\*; import java.net.\*; import java.lang.\*; public class RCEclient

{

public static void main(String a[])throws UnknownHostException,IOException

{

Socket c=new Socket("localhost",1680); System.out.println("Enter the command:");

DataInputStream dis=new DataInputStream(System.in); String s=dis.readLine();

OutputStream os=c.getOutputStream(); DataOutputStream dos=new DataOutputStream(os);

dos.write(s.getBytes()); dos.close();

dis.close();

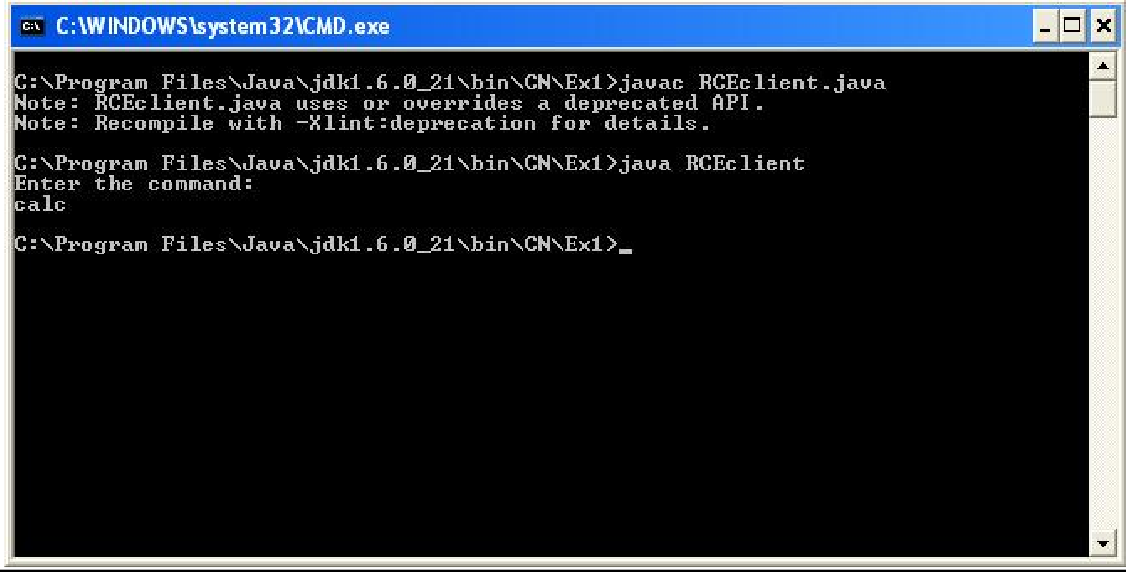
c.close();

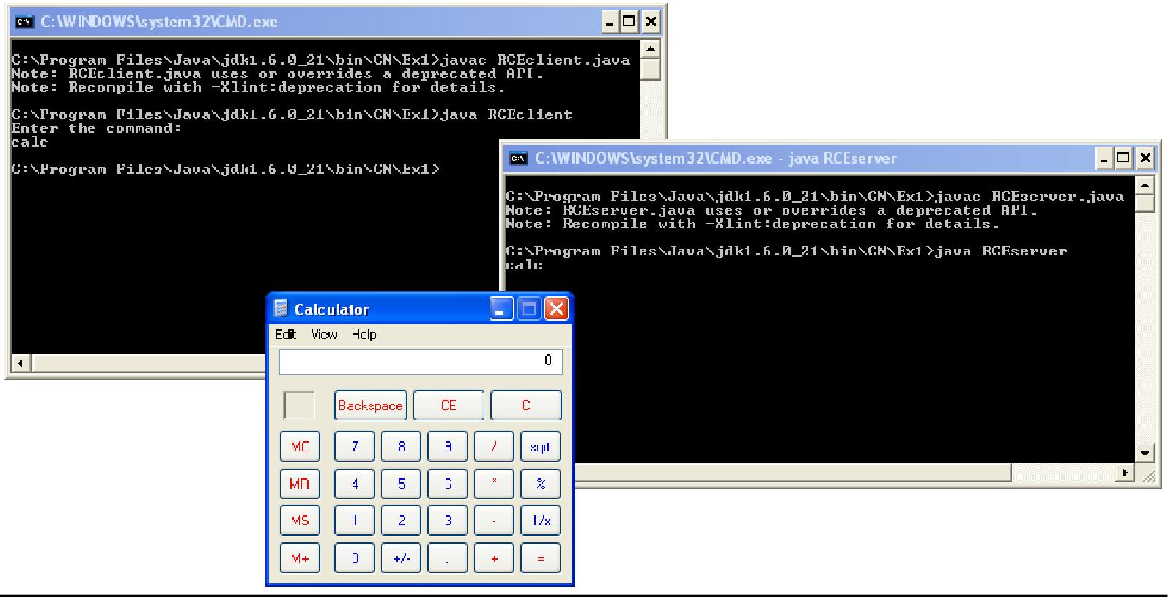
}

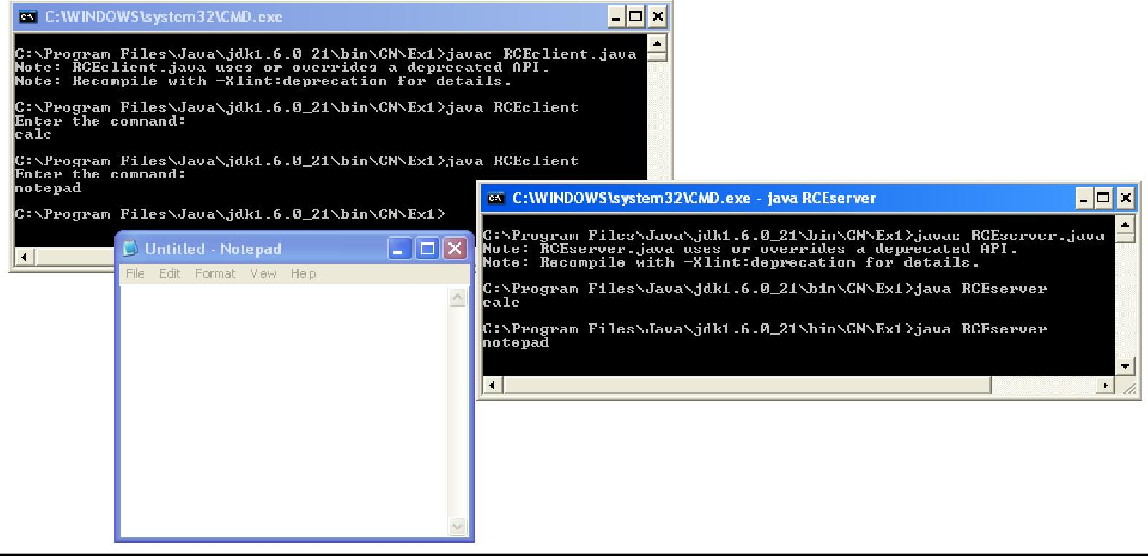
}

# Output: RCEserver.java

**RCEclient.java**







**Result:**

Thus the program remote command execution (calculator and notepad) executed andverified.

|  |  |
| --- | --- |
| **Ex. No:4.1** | **ONE WAY COMMUNICATION USING TCP** |
| **Date :** |

# Aim:

To write a java program for one way communication using TCP. **Algorithm:**

# Server Program

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called “Tserver1”.
4. Within class “Tserver1”, create a ServerSocket object called “ss” with the port number8000.
5. Create a Socket object called “s” by using the accept() method, which listens for a connection to be made to this server socket and acceptsit.
6. CreateanewDataInputStreamobject,whichactsasaninputstreamtotheserverfromtheclient.
7. Create a new PrintStream object, which acts as an output stream to the client from theserver.
8. Display the message “Server ready...” on the serverwindow.
9. Repeat the followingsteps:
   1. Prompt for the message to be sent from server toclient.
   2. Read in the message to be sent from theuser.
   3. Send the message to the client using the PrintStreamobject.
   4. If the message equals the string “end”, then close the input and output streams and exit the loop.
10. Stop **ClientProgram**
11. Start.
12. Import the java.net, java.io, and java.utilpackages.
13. Create a new class called“Tclient1”.
14. Inside “Tclient1”, create a new Socket object called “s” with port number8000.
15. CreateanewDataInputStreamobjectwhichactsastheinputstreamtotheclientfromtheserver.
16. Repeat the followingsteps:
    1. Read in the input from the server to the client using the DataInputStreamobject.
    2. Display the messagereceived.
    3. Ifthestringreceivedequals“end”,thenclosetheinputandoutputstreamsandexittheloop.
17. Stop.

**TServer1.java** import java.io.\*; import java.net.\*; class TServer1

{

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

{

ServerSocket ss=new ServerSocket(8000);//implements server sockets

Socket s=ss.accept();//Listens for a connection to be made to this socket and accepts it DataInputStream in=new DataInputStream(System.in);

PrintStream dos=new PrintStream(s.getOutputStream()); System.out.println("Server ready....!");

while(true)

{

System.out.println("Enter message to send:"); String str=in.readLine();

dos.println(str);

if(str.equals("end"))

{

ss.close(); break;

}

}

}

}

# TClient1.java

import java.io.\*; import java.net.\*; classTClient1

{

public static void main(String args[]) throws IOException

{

Socket s=new Socket("localHost",8000);// implements client sockets DataInputStream in=new DataInputStream(s.getInputStream());

while(true)

{

String str=in.readLine();

System.out.println("Message Received from Server:"+str);

if(str.equals("end"))

{

s.close(); break;

}

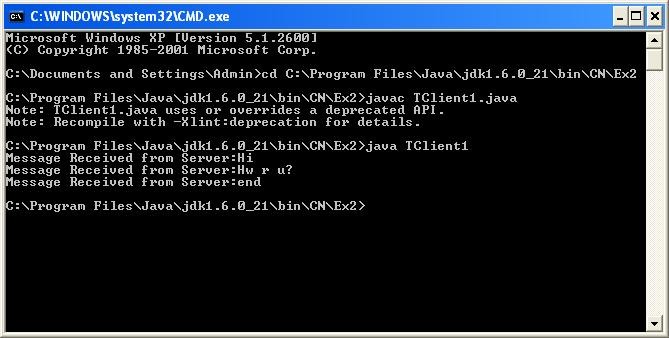
}

}

}

# Output: TServer1.java

**TClient1.java**



**Result:**

Thus the program for one way communication using TCP executed and verified.

|  |  |
| --- | --- |
| **Ex. No:4.2** | **TWO WAY COMMUNICATION USING TCP** |
| **Date :** |

# Aim:

To write a java program for two way communication using TCP. **Algorithm**

# Server Program

1. Start.
2. Import the java.net and java.iopackages.
3. Create a new class called“Tserver2”.
4. Inside class “Tserver2”, create a new ServerSocket object called “ss” with the port number 5555.
5. Create a new Socket object called “s” by using the accept() method, which listens for a connection to be made to this server socket and acceptsit.
6. Create a new PrintStream object which acts as an output stream from the server to the client.
7. CreateanewDataInputStreamobjectwhichactsasaninputstreamfromtheclienttotheserver.
8. Display the prompt “Server isready”.
9. Repeat the followingsteps:
   1. Prompt and read in the message from the user to be sent to the client from theserver.
   2. Send the message to the client using the PrintStreamobject.
   3. If the message equals the string “end”, then close the input and output streams and exit the loop.
   4. Else, read in the message from the client using the DataInputStreamobject.
   5. Display the message received from the client to theuser.
   6. If the message received equals the string “end”, then close the input and output streams and exit theloop.
10. Stop.

# Client Program

1. Start.
2. Import the java.net and java.iopackages.
3. Create a new class called“Tclient2”.
4. Inside “Tclient2”, create a new Socket object called “s” with the port number5555.
5. CreateanewDataInputStreamobjectwhichactsasaninputstreamtotheclientfromtheserver.
6. Create a new PrintStream object which acts as an output stream to server from theclient.
7. Repeat the followingsteps:
   1. Read in the input from the server using the DataInputStreamobject.
   2. Display the message received from theserver.
   3. If the message received equals “end”, then close input & output streams and exitloop.
   4. Else, Prompt and read in the message from the user, which is to be sent to theserver.
   5. Send the message to the server from the client using the PrintStreamobject.
   6. If the message equals “end”, then close the input and output streams and exitloop.
8. Stop.

# TServer2.java

import java.io.\*; import java.net.\*; classTServer2

{

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

{

ServerSocket ss=new ServerSocket(5555); Socket s=ss.accept();

PrintStream dos=new PrintStream(s.getOutputStream()); DataInputStream in=new DataInputStream(System.in); DataInputStream inn=new DataInputStream(s.getInputStream()); System.out.println("Server Ready!! ");

while(true)

{

System.out.println("Enter the msg to send: "); String str=in.readLine();

dos.println(str); if(str.equals("end"))

{

ss.close(); break;

}

String str1=inn.readLine(); System.out.println("Msg received :"+str1);

if(str1.equals("end"))

{

ss.close(); break;

}

}

}

}

# TClient2.java

import java.io.\*; import java.net.\*; classTClient2

{

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

{

Socket s=new Socket("LocalHost",5555);

DataInputStream in=new DataInputStream(s.getInputStream()); DataInputStream inn=new DataInputStream(System.in); PrintStream dos=new PrintStream(s.getOutputStream());

while(true)

{

String str=in.readLine(); System.out.println("Message received :"+str);

if(str.equals("end"))

{

s.close(); break;

}

System.out.println("Enter the msg to send : "); String str1=inn.readLine();

dos.println(str1);

if(str1.equals("end"))

{

s.close(); break;

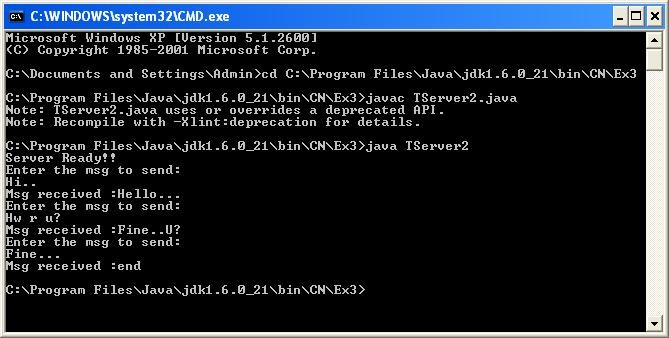
}

}

}

}

# Output: TServer2.java



**TClient2.java**



**Result:**

Thus the program for two way communication using TCP executed and verified.

|  |  |
| --- | --- |
| **Ex. No:4.3** | **ECHO COMMAND** |
| **Date :** |

# Aim:

To implement echo server and client in java using TCP sockets.

# Algorithm: Server

# program

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called “EchoServer”.
4. Inside “EchoServer”, declare a ServerSocket class object called “ss” with port number5362.
5. Display“Echo-Server”.
6. Create a new Socket object called “s” using the accept()method.
7. CreateanewDataInputStreamobjectwhichactsasaninputstreamfromtheclienttotheserver.
8. Create a new PrintStream object which acts as an output stream to the client from theserver.
9. Repeat the following steps:
   1. Read in the message from the client using the DataInputStreamobject.
   2. If the received message is equal to the string “end” then close the input and output streams and exit theprogram.
   3. Else, display the message received from the client to the serverconsole.
   4. Send the same message back to the client using the PrintStreamobject.
   5. Display “Message sentsuccessfully.”
10. Stop. **ClientProgram**
11. Start.
12. Import the java.net and java.iopackages.
13. Declare a new class called“EchoClient”.
14. Inside “EchoClient”, create a new Socket object called “s” with the port number5362.
15. Create a new BufferedReader object that acts as an input stream to the client from theserver.
16. Create another BufferedReader object that acts as an input stream from the user to theclient.
17. Create a new PrintWriter object that acts as an output stream from the client to theserver.
18. Create a new String object called“str”.
19. Display“Echo-Client”.
20. Repeat the followingsteps:
    1. Prompt and read in the message to be sent to the server, from the user, using the appropriate BufferedReader object and store in“str”.
    2. Send the message stored in “str” to the server using the PrintWriterobject.
    3. Flush the output stream to theserver.
    4. If “str” is equal to the string “end”, then close the input and output streams and exit the program.
    5. Else, display “Message sentsuccessfully”.
    6. Read in the message echoed back from the server to the client using the appropriate BufferedReaderobject.
    7. Display the message echoed back from theserver.
21. Stop.

# EchoServer.java

import java.io.\*; importjava.net.\*;

public class EchoServer

{

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

{

String str;

ServerSocket ss=new ServerSocket(5362); System.out.println("Echo-Server"); System.out.println("-------------------"); Socket s=ss.accept();

DataInputStream dis=new DataInputStream(s.getInputStream()); PrintStream ps=new PrintStream(s.getOutputStream());

while(true)

{

str=dis.readLine(); if(str.equals("end"))

{

dis.close();

ps.close(); System.exit(0);

}

System.out.println("Message Received from Client : "+str); ps.println(str);

ps.flush();

System.out.println("Message sent successfully!");

}

}

}

# EchoClient.java

import java.io.\*; import java.net.\*; public classEchoClient

{

public static void main(String args[])

{

try

{

Socket s=new Socket("LocalHost",5362);

BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream())); BufferedReader brUser=new BufferedReader(new InputStreamReader(System.in)); PrintWriter pw=new PrintWriter(s.getOutputStream());

String str; System.out.println("Echo-Client"); System.out.println("--------------");

while(true)

{

System.out.println("\nSend message to server: "); str=brUser.readLine();

pw.println(str); pw.flush();

if(str.equals("end"))

{

if(br!=null) br.close(); if(pw!=null) pw.close(); System.exit(0);

}

System.out.println("Message sent successfully!"); System.out.println("Echo Message from server:"+br.readLine());

}

}

catch(IOException e)

{

System.err.println(e);

}

}

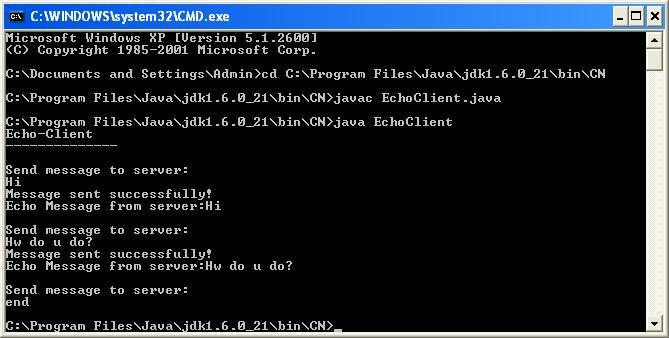
}

# Output:

**EchoServer.java**



# EchoClient.java



**Result:**

Thus data from client to server is echoed back to the client executed and verified.

|  |  |
| --- | --- |
| **Ex. No :4.4** | **PING COMMAND** |
| **Date :** |

# Aim:

To implement java program for check connection between client and server using ping

command. **Algorithm:**

**Server program**

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called“PingServer”.
4. Inside “PingServer”, declare a ServerSocket object called “ss” with port number6523.
5. Create a new String object called“str”.
6. Create a new Socket object called “s” using the accept()method.
7. CreateanewDataInputStreamobjectwhichactsasaninputstreamfromtheclienttotheserver.
8. Create a new PrintStream object which acts as an output stream to the client from theserver.
9. Display “PingServer”.
10. Repeat the following steps for integer variable i=0 , 1, …,4
    1. Read in the message from the client using the DataInputStream object and store in“str”.
    2. Display “Pinged byclient”.
    3. Send the message “bytes=32 time<1ms TTL=128” to the client using the PrintStream object.
11. Close the input and output streams and close the ServerSocketobject.
12. Stop

# Client Program

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called“PingClient”.
4. Inside “PingClient”, declare the integer variables “i”, “j”,“k”.
5. Create a DataInputStream object that acts as an input stream from the user to theclient.
6. Prompt and read in the IP address from theuser.
7. CreateanewSocketobjectcalled“s”withtheIPaddressfromtheuserandportnumber5362.
8. CreateanewBufferedReaderobjectthatactsasaninputstreamtotheclientfromtheserver.
9. CreateanewPrintWriterobjectthatactsasanoutputstreamfromtheclienttotheserver.
10. Display “Pinging with byte=32bytes ofdata.”
11. Create a new String object called“str”.
12. Repeat the following steps for i=0 , 1, …,4
    1. Send the IP address received from the user to the server using the PrintWriterobject.
    2. Flush the output stream to theserver.
    3. Read in the message from the server and store in“str”.
    4. If str does not equal to null, then display the message from the server with atime delay.
    5. Else, display “Request Timed Out!” with timedelay.
13. Close the input and output streams and close the socketobject.
14. Stop

# Pingserver.java

import java.io.\*; import java.net.\*; public classPingserver

{

public static void main(String args[])

{

try

{

String str;

ServerSocket ss=new ServerSocket(6523); Socket s=ss.accept();//accept socket

DataInputStream dis=new DataInputStream(s.getInputStream()); PrintStream ps=new PrintStream(s.getOutputStream());//printdata System.out.println("Ping Server");

System.out.println("----------------"); for(int i=0;i<4;i++)

{

str=dis.readLine();//read the received data System.out.println("Pinged By Client!");//input message is copied

ps.println(str+": bytes=32 time<1ms TTL=128");//message sent to client

}

dis.close();

ps.close();

ss.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

# Pingclient.java

import java.io.\*; import java.net.\*; public classPingclient{

public static void main(String args[]) throws Exception{ int i,j,k;

System.out.println("Ping Client"); System.out.println("----------------");

System.out.print("Enter the IP address:");// to get IP Address DataInputStream disUser=new DataInputStream(System.in); String ip=disUser.readLine();

Socket s=new Socket(ip,6523); //establish connection

BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream())); PrintWriter pw=new PrintWriter(s.getOutputStream()); //Output stream that has print(), println() System.out.println("\n Pinging ["+ip+"] with 32 bytes of data:\n");

String str; for(i=0;i<4;i++)

{

pw.println(ip); //send data to socket pw.flush();

str=br.readLine(); if(str!=null){ for(j=0;j<20000;j++)

{ for(k=0;k<50000;k++);

}

System.out.println("Reply from "+str);

}

else

{

for(i=0;i<4;i++)

{ for(j=0;j<10000;j++)

{ for(k=0;k<50000;k++);

}

System.out.println("\n Request Timed Out!");

}}}

if(br!=null) br.close(); if(pw!=null) pw.close();

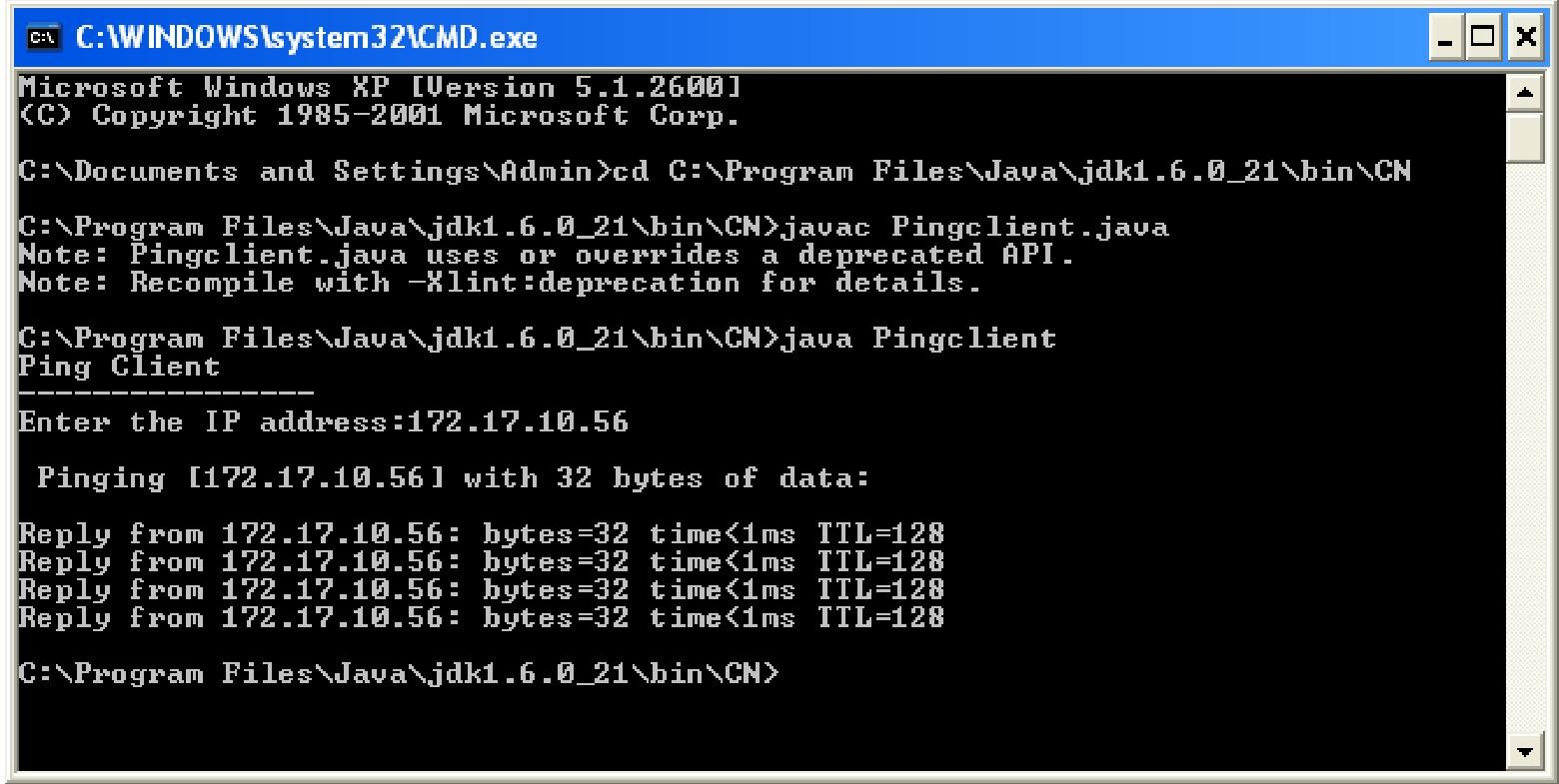
* 1. lose();

}

}

# Output: Pingserver.java

**Pingclient.java**



**Result:**

Thus the program for connection availability between client and server using ping commandis executed and verified.

|  |  |
| --- | --- |
| **Ex. No:4.5** | **TALK COMMAND** |
| **Date:** |

# Aim:

To implement java program for message passing between client and server using talk command.

# Algorithm: Server Program

* + 1. Start.
    2. Import the java.net and java.iopackages.
    3. Declare a new class called“TalkServer”.
    4. Inside “TalkServer( )” constructor, declare a ServerSocket object called “ss” with port number 3453.
    5. Display “Talk Server” and “Server islistening...”.
    6. Create a new Socket object called “s” using the accept()method.
    7. Create a new PrintStream object which acts as an output stream to the client from theserver.
    8. Create a new BufferedReader object that acts as an input stream to the server from theclient.
    9. CreateanotherBufferedReaderobjectwhichactsasaninputstreamfromtheusertotheserver.
    10. Create a new String object called“str”.
    11. Repeat the followingsteps:
        1. ReadinthemessagefromtheclientusingthecorrectBufferedReaderobjectandstorein“str”.
        2. Display the message received from the client in“str”.
        3. If string in “str” equals “end” while ignoring case, then close the input and output streams, close the ServerSocket object, and exit theprogram.
        4. Else, Prompt and read in the message to send to the client from the user using the correct BufferedReader object and store in“str”.
        5. Send the message in “str” to the client using the PrintStreamobject.
        6. If string in “str” equals “end” while ignoring case, then close the input and output streams, close the ServerSocket object, and exit theprogram.
    12. In the main() method, create a new object of TalkServer and callthe TalkServer() constructor.
    13. Stop.

# Client Program

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called “TalkClient”.
4. Inside “TalkClient()” constructor, create a new Socket object called “s” with the port number 3453.
5. Display “TalkClient”.
6. Create a new BufferedReader object that acts as an input stream from the user to theclient.
7. CreateanewPrintStreamobjectthatactsasanoutputstreamfromtheclienttotheserver.
8. CreateanewBufferedReaderobjectthatactsasaninputstreamtotheclientfromtheserver.
9. Create a new String object called“str”.
10. Repeat the followingsteps:
    1. Prompt and read in the message to be sent to the server, from the user, using the appropriate BufferedReader object and store in“str”.
    2. Send the message stored in “str” to the server using the PrintStreamobject.
    3. If string in “str” equals “end” while ignoring case, then close the input and output streams, close the Socket object, and exit theprogram.
    4. Read in the messagefrom the server and store in “str” using correct BufferedReader object.
    5. Display the message received from theserver.
    6. If string in “str” equals “end” while ignoring case, then close the input and output streams, close the Socket object, and exit theprogram.
11. In the main() method, create a new object of TalkClient and call the TalkClient()constructor.
12. Stop.

# TalkServer.java

import java.io.\*; importjava.net.\*;

public class TalkServer{ TalkServer(){

try

{

ServerSocket ss = new ServerSocket(3453); System.out.println("Talk Server\nServer is listening..."); Socket s = ss.accept();

PrintStream ps = new PrintStream(s.getOutputStream());

BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream())); BufferedReader brUser = new BufferedReader(new InputStreamReader(System.in)); String str;

while(true)

{

str = br.readLine(); System.out.println("Message received: "+ str); if(str.equalsIgnoreCase("end"))

{

ps.close();

br.close();

ss.close();

s.close(); System.exit(0);

}

System.out.println("Enter the message to send: "); str = brUser.readLine();

ps.println(str); if(str.equalsIgnoreCase("end"))

{

ps.close();

br.close();

ss.close();

s.close(); System.exit(0);

}}}

catch(Exception e)

{

System.err.println(e);

}}

public static void main(String a[])

{

new TalkServer();

}}

# TalkClient.java

import java.io.\*; import java.net.\*; public classTalkClient

{

TalkClient()

{

Try

{

System.out.println("Talk Client");

Socket s = new Socket("LocalHost",3453);

BufferedReader brUser = new BufferedReader(new InputStreamReader(System.in)); PrintStream ps = new PrintStream(s.getOutputStream());

BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream())); String str;

while(true)

{

System.out.println("Enter the message to send: "); str = brUser.readLine();

ps.println(str); if(str.equalsIgnoreCase("end"))

{

ps.close();

br.close();

s.close(); System.exit(0);

}

str = br.readLine(); System.out.println("Message received: " + str); if(str.equalsIgnoreCase("end"))

{

ps.close();

br.close();

* 1. lose(); System.exit(0);

}}}

catch(Exception e)

{

System.err.println(e);

}

}

public static void main(String a[])

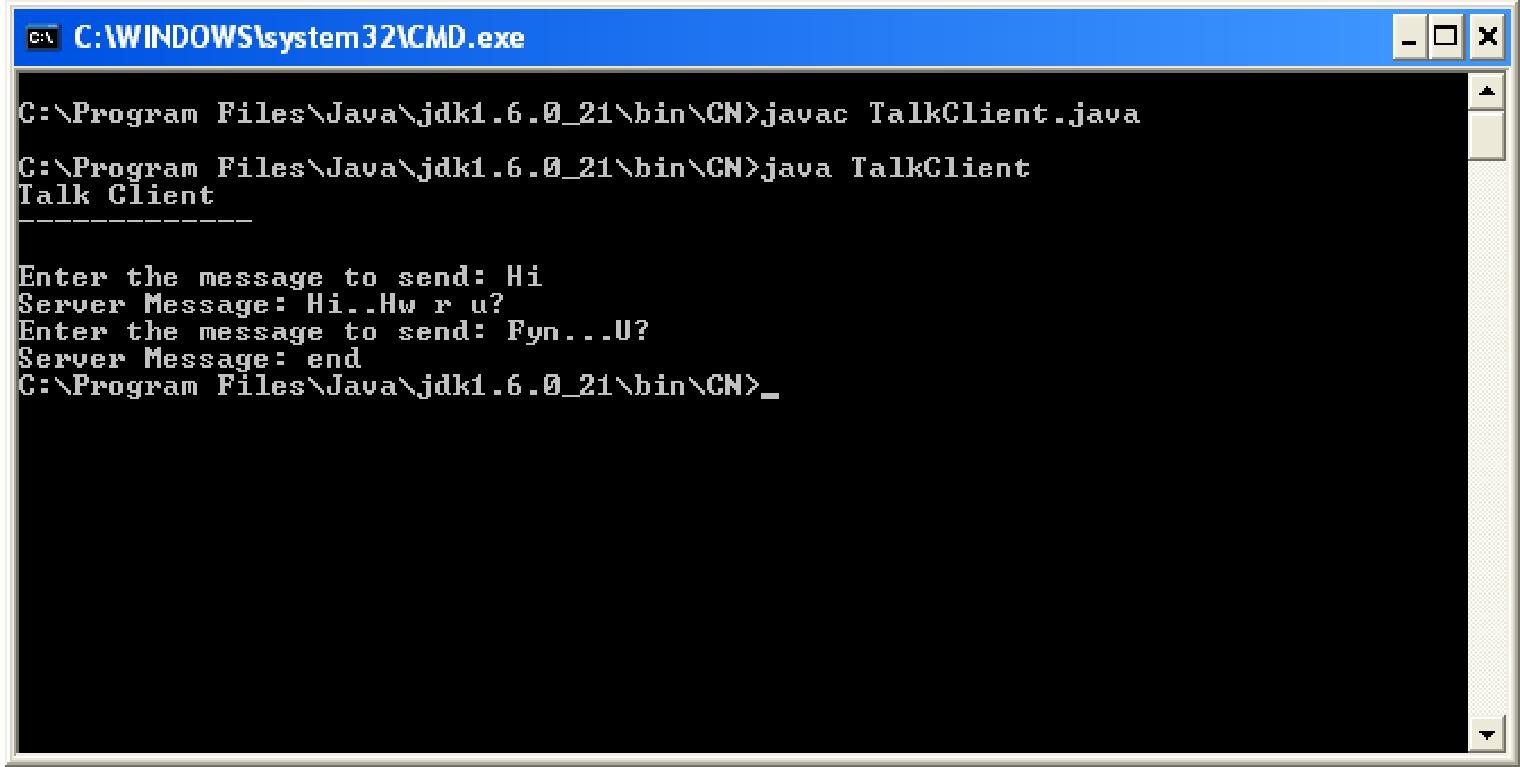
{

new TalkClient();

}}

# Output: TalkServer.java

**TalkClient.java**



**Result:**

Thus the program for message passing between client and server using talk command is executed and verified.

|  |  |
| --- | --- |
| **Ex. No:4.6** | **FILE TRANSFER USING TCP** |
| **Date:** |

# Aim:

To implement a java program for file transfer between two nodes using TCP.

# Algorithm: Server Program

* + 1. Start.
    2. Import the java.net and java.iopackages.
    3. Declare a new class called “FileServer”.
    4. Inside “FileServer”, declare a ServerSocket object called “ss” with port number3891.
    5. Declare a new bytearray
    6. Repeat the followingsteps:
       1. Display “Server islistening...”
       2. Create a new Socket object called “s” using the accept()method.
       3. Display “New connectionaccepted”.
       4. Declare and initialize integer variables “len” and “bytecount” tozero.
       5. Create another BufferedReader object which acts as an input stream from the user to the server.
          1. Create a new String object called“str”.
       6. Prompt and read in the file name to be saved from the user and store in“str”.
       7. Create a FileOutputStream object using the file name in“str”.
       8. Create a BufferedInputStream object with byte array size of 1024, which acts as an input stream to the server from theclient.
       9. Repeat the following steps for each character read from the client and stored in byte array, using theBufferedInputStream:
          1. Write each character in byte array to file usingFileOutputStream.
          2. Increment “bytecount” by1.
       10. Display the no. of Bytes written in“bytecount”.
       11. Close the input and output streams and close the Socketobject.
       12. Prompt and read in if the user wants toexit.
       13. If “str” equals “y” while ignoring case, then close the ServerSocket object and exit the program.
    7. Stop

# Client Program

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called “FileClient”.
4. Inside “FileClient”, create a new Socket object called “s” with the port number3891.
5. Declare a new bytearray.
6. Create a new BufferedOutputStream object that acts as an output stream from the client to the server.
7. Create a new BufferedReader object that acts as an input stream to the client from theuser.
8. Create a new String object called“str”.
9. Prompt and read in the name of the file to be sent to the server, from the user, using BufferedReader object and store in“str”.
10. Prompt and read in if the user wants to view the contents of the file, using Buffered Reader object, and store in new String“c”.
11. Create a new FileInputStream object that acts as an input stream from the file to theclient.
12. Create a new File object using the file name stored in“str”.
13. If string “c” equals letter “y” while ignoring case, then read in each character from the file, store in byte array “b”, and display “b” to the user while the no. of characters read is less than 800 or the end of file isreached.
14. Close and create the FileInputStream object again to reset the position of thestream.
15. Declare and initialize variables “len” and “bytecount”, of type long, tozero.
16. Assign the file size to“len”.
17. Display the length of the file to be transferred which is stored in“len”.
18. Repeat the following steps for each character read from the file and stored in the byte array, using the FileInputStreamobject:
    1. Increment “bytecount” by1.
    2. Write each character in byte array to the server using the BufferedOutputStreamobject.
    3. Flush the output stream from the client to theserver.
19. Display the no. of bytes sent which is stored in“bytecount”.
20. Close the input and output streams and close the Socketobject.
21. Stop.

# FileServer.java

import java.net.\*; importjava.io.\*;

public class FileServer {

public static void main(String args[])

{

try {

ServerSocket ss = new ServerSocket(3891); byte b[] = new byte[1024];

System.out.println("\nServer is listening on port "+ss.getLocalPort()+"..."); while(true)

{

Socket s = ss.accept();

System.out.println("New connection accepted " + s.getInetAddress() + ":" + s.getPort()); int len=0, bytecount=0;

System.out.println("Enter the name of the file to be saved: ");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); String str = br.readLine();

FileOutputStream fos = new FileOutputStream(str);

BufferedInputStream bis = new BufferedInputStream(s.getInputStream(),1024); while( (len = bis.read(b,0,1)) != -1)

{

fos.write(b,0,1); bytecount++;

}

System.out.println("Bytes Written: "+bytecount); bis.close();

fos.close();

s.close();

System.out.print("Press y if you want to exit: "); str = br.readLine(); if(str.equalsIgnoreCase("y"))

{

ss.close(); System.exit(0);

}}}

catch(Exception e)

{

System.err.println(e);

}}}

# FileClient.java

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

public class FileClient {

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

Socket s = new Socket("LocalHost", 3891);

System.out.println("Connected with server "+s.getInetAddress()+ ":"+s.getPort()); byte b[] = new byte[324];

BufferedOutputStream bos = new BufferedOutputStream(s.getOutputStream(),1024); System.out.println("Enter the file name to send: ");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); String str = br.readLine();

System.out.print("Press y if you want to view the file's contents: "); String c = br.readLine();

FileInputStream fis = new FileInputStream(str); File f = new File(str); if(c.equalsIgnoreCase("y")){

int total = 0, nRead = 0;

while((nRead = fis.read(b)) != -1 && total<800){ System.out.println(new String(b));

total += nRead; } if(total<f.length()){ System.out.println("...");

}

fis.close();

fis = new FileInputStream(str);

}

long len, bytecount=0; len=f.length();

System.out.println("\nLength of the file to be transferred: " + len); while((len=fis.read(b,0,1)) != -1){

bytecount++; bos.write(b,0,1); bos.flush();

}

System.out.println("Bytes Sent: "+bytecount); fis.close();

bos.close();

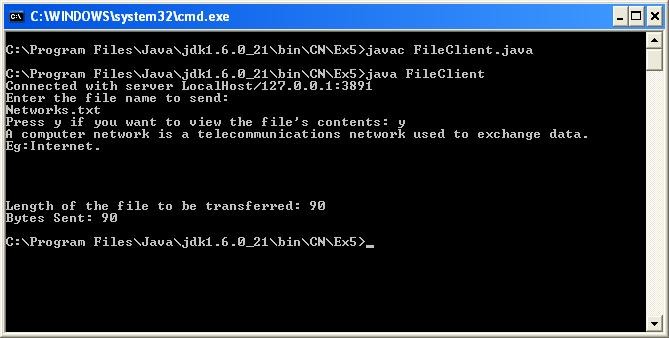
* 1. lose(); System.exit(0);

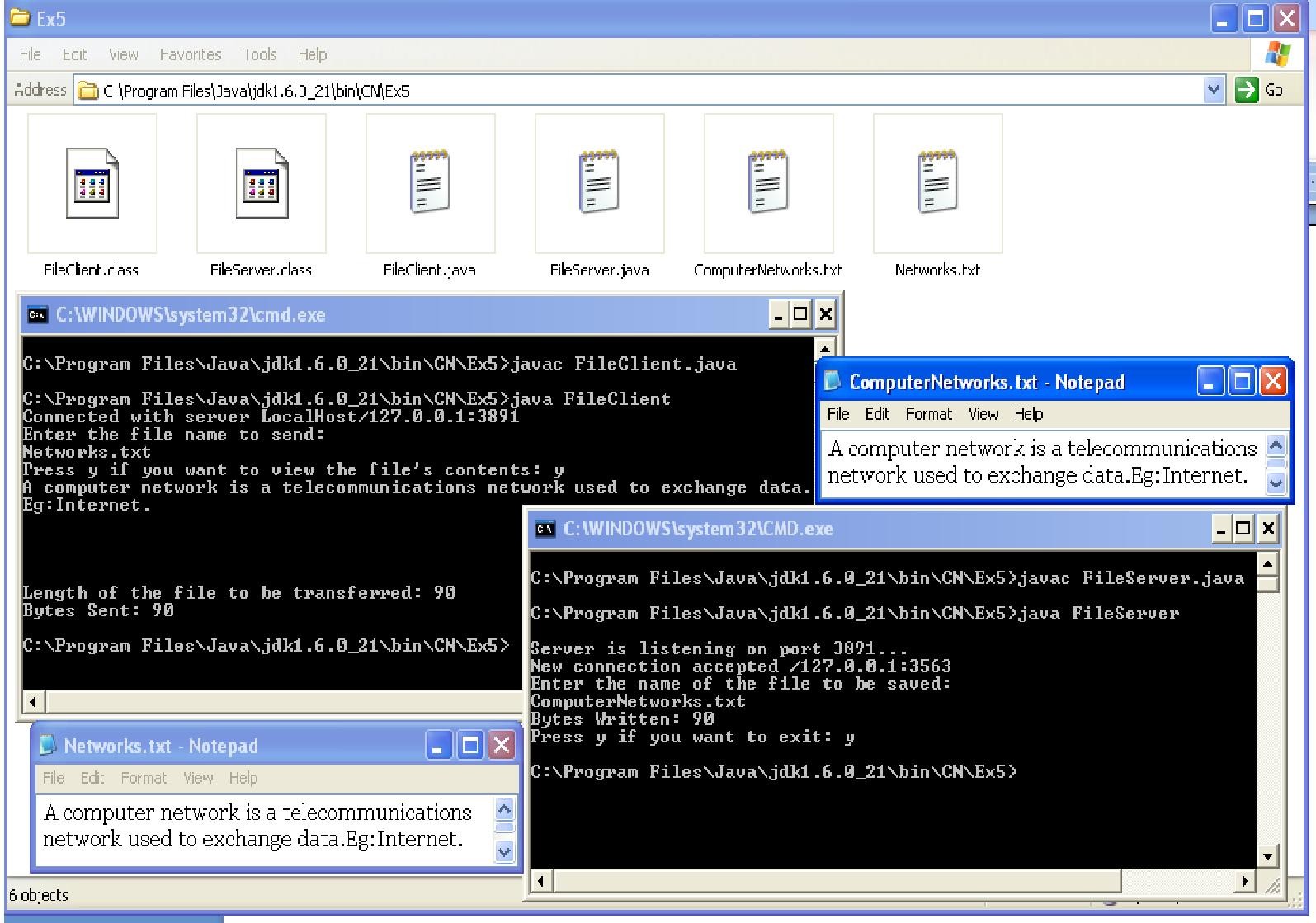
}

catch(Exception e){ System.err.println(e);}}}

# Output: FileServer.java

**FileClient.java**





**Result:**

Thus the program for file transfer between two nodes using TCP is executed and verified.

|  |  |
| --- | --- |
| **Ex. No:5.1** | **ONE WAY COMMUNICATION USING UDP** |
| **Date:** |

# Aim:

To implement a java program for one way communication using UDP. **Algorithm:**

# Server Program

* + 1. Start.
    2. Import the java.netpackage.
    3. Declare a new class called“UServer1”.
    4. Inside class “UServer1”, create a new DatagramSocket object called “ds” using the port number 9875.
    5. Display “UDP Server” and “Serverready...!”.
    6. Repeat the followingsteps:
       1. Create a byte array “b” of size 1024bytes.
       2. CreateanewDatagramPacketcalled“dp”,passingthebytearrayanditslengthasarguments.
       3. Receive the DatagramPacket sent from the client and store it in “dp” using the DatagramSocket object.
       4. Using the getData() method, retrieve the strings encapsulated by the packet and store them in “str”.
       5. Display the message received from the client which is stored in string“str”.
       6. If string “str” equals “end” while ignoring case, then break theloop.
    7. Close the DatagramSocket object and exit theprogam.
    8. Stop.

# Client Program

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called “UClient1”.
4. Inside “UClient1”, create a new BufferedReader object that acts as an input stream from the user to theclient.
5. Create a new DatagramSocketobject.
6. Get the Local Host Inet Address and store it in an InetAddress object called“IPAddress”.
7. Display “UDPClient”.
8. Repeat the followingsteps:
   1. Create a new byte array “b” of size1024.
   2. Prompt and read in the message to send to the server and store in string“str”.
   3. Get the bytes of string “str” and store in byte array“b”.
   4. Create a new DatagramPacket object using the byte array “b” and its length, IPAddress and port number9875.
   5. Send the DatagramPacket object to the server using the DatagramSocketobject.
   6. If string “str” equals “end” while ignoring case, then close the DatagramSocket object and exit theprogram.
9. Stop.

# UServer1.java

import java.net.\*; class UServer1 {

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

DatagramSocket ds = new DatagramSocket(9875); //Server Socket Created System.out.println("UDP Server");

System.out.println("--------------"); System.out.println("Server Ready...!"); while(true){

byte[] b = new byte[1024];

DatagramPacket dp = new DatagramPacket(b,b.length); ds.receive(dp);

String str = new String(dp.getData(),0,0,dp.getLength()); System.out.print("\nMessage Received from Client: " + str); if(str.equalsIgnoreCase("end")){

break;

}}

ds.close(); System.exit(0);

}}

# UClient1.java

import java.io.\*; import java.net.\*; class UClient1 {

public static void main(String args[]) throws Exception,BindException{ BufferedReader br =new BufferedReader(new InputStreamReader(System.in)); DatagramSocket ds = new DatagramSocket(); //Client Socket is created

InetAddress IPAddress = InetAddress.getByName("localhost"); //Gets the IP Address System.out.println("UDP Client");

System.out.println("--------------"); while(true){

byte[] b = new byte[1024]; System.out.print("Enter the message to server: "); String str = br.readLine();

b = str.getBytes(); //sends data

DatagramPacket dp = new DatagramPacket(b, b.length, IPAddress, 9875); ds.send(dp);

if(str.equalsIgnoreCase("end")){ ds.close();

System.exit(0);

}}}}

# Output: UServer1.java

**UClient1.java**



**Result:**

Thus the program for one way communication using UDP is executed and verified.

|  |  |
| --- | --- |
| **Ex. No:5.2** | **TWO WAY COMMUNICATION USING UDP** |
| **Date:** |

# Aim:

To implement a java program for two way communication using UDP.

# Algorithm: Server Program

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called“UServer2”.
4. Create a new DatagramSocket object called “ds” using the port number9876.
5. Create a new BufferedReader object which acts asan input stream from the user to the server.
6. Repeat the followingsteps:
   1. Create two byte arrays “rcvData” and “sndData”, each of size1024.
   2. Create a new DatagramPacket object called “rcvPacket” using the byte array “rcvData” and its length.
   3. Using the DatagramSocket object, receive the DatagramPacket from the client and store in “rcvPacket”.
   4. Get the string data that is encapsulated within “rcvPacket” and store it in String object“str”.
   5. Display the string value in“str”.
   6. If string “str” equals “end” while ignoring case, then close the DatagramSocket object and exit theprogram.
   7. Prompt and read in the message to send to client and store in“str”.
   8. Get the equivalent bytes of string object “str” and store in byte array“sndData”.
   9. Create a new DatagramPacket “sndPacket” using the byte array “sndData” and its length, and the Inet Address and port number of“rcvPacket”.
   10. Send “sndPacket” to the client using the DatagramSocketobject.
   11. If string “str” equals “end” while ignoring case, then close the DatagramSocket object and exit theprogram.
7. Stop

# Client Program

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a new class called “UClient2”.
4. Create a new DatagramSocketobject.
5. Create a new BufferedReader object that acts as an input stream from the user to theclient.
6. Get the local hostInetAddress.
7. Repeat the followingsteps:
   1. Create two byte arrays “rcvData” and “sndData”, each of size1024.
   2. Prompt and read in the message to send to the server and store in“str”.
   3. Get the equivalent bytes of string object “str” and store in byte array“sndData”.
   4. Create a new DatagramPacket “sndPacket” using the byte array “sndData” and its length, and the Inet Address and port number9876.
   5. Send “sndPacket” to the server using the DatagramSocketobject.
   6. If string “str” equals “end” while ignoring case, then close the DatagramSocket object and exit theprogram.
   7. Create a new DatagramPacket object called “rcvPacket” using the byte array “rcvData” and its length.
   8. Using the DatagramSocket object, receive the DatagramPacket from the server and store in “rcvPacket”.
   9. Get the string data that is encapsulated within “rcvPacket” and store it in String object“str”.
   10. Display the string value in“str”.
   11. If string “str” equals “end” while ignoring case, then close the DatagramSocket object and exit theprogram.
8. Stop.

# UServer2.java

import java.io.\*; import java.net.\*; classUServer2

{

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

{

System.out.println("UDP SERVER"); System.out.println("--------------"); DatagramSocket ds = newDatagramSocket(9876);

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

{

byte[] rcvData = new byte[1024]; byte[] sndData = new byte[1024];

DatagramPacket rcvPacket = new DatagramPacket(rcvData,rcvData.length); ds.receive(rcvPacket);

String str = new String(rcvPacket.getData(),0,0,rcvPacket.getLength()); System.out.print("Message Received From Client: " + str); if(str.equalsIgnoreCase("end"))

{

ds.close(); System.exit(0);

}

System.out.print("\nEnter the Message to send to Client: "); str = br.readLine();

InetAddress IPAddress = rcvPacket.getAddress(); int port = rcvPacket.getPort();

sndData = str.getBytes();

DatagramPacket sndPacket = new DatagramPacket(sndData, sndData.length, IPAddress, port); ds.send(sndPacket);

if(str.equalsIgnoreCase("end"))

{

ds.close(); System.exit(0);

}

}

}

}

# UClient2.java

import java.io.\*; import java.net.\*; classUClient2

{

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

{

System.out.println("UDP CLIENT"); System.out.println("--------------"); DatagramSocket ds = new DatagramSocket();

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); InetAddress IPAddress = InetAddress.getByName("localhost");

while(true)

{

byte[] sndData = new byte[1024]; byte[] rcvData = new byte[1024];

System.out.print("Enter the Message to send to Server: "); String str = br.readLine();

sndData = str.getBytes();

DatagramPacket sndPacket = new DatagramPacket(sndData, sndData.length,IPAddress, 9876); ds.send(sndPacket);

if(str.equalsIgnoreCase("end"))

{

ds.close(); System.exit(0);

}

DatagramPacket rcvPacket = new DatagramPacket(rcvData, rcvData.length); ds.receive(rcvPacket);

str = new String(rcvPacket.getData(),0,0,rcvPacket.getLength()); System.out.println("Message received from server: " + str); if(str.equalsIgnoreCase("end"))

{

ds.close(); System.exit(0);

}

}

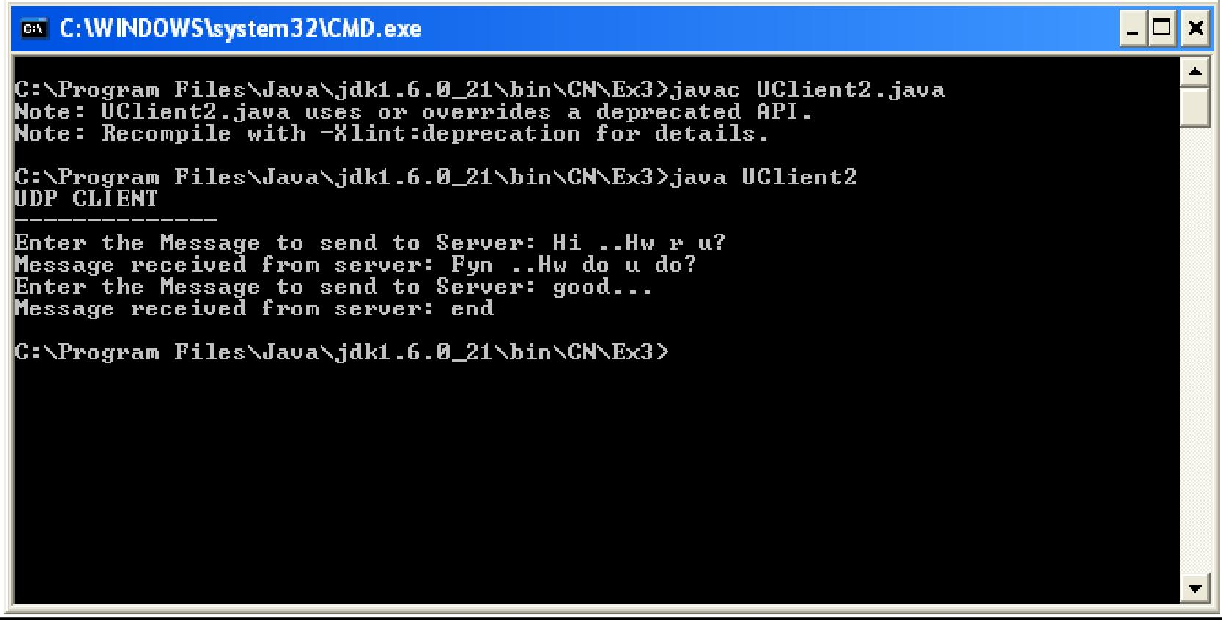
}

}

# Output: UServer2.java



**UClient2.java**



**Result:**

Thus the program for two way communication using UDP is executed and verified.

|  |  |
| --- | --- |
| **Ex. No: 6** | **REMOTE METHOD INVOCATION** |
| **Date:** |

# Aim:

To implement a java for simple RMI programming **Algorithm:**

1. Start theprogram.
2. Establishtheconnectionbetweentheclientandtheservertocalculatethefactorialoperation.
3. In implement() it calls the inter() which throws remote exception thrown byfact().
4. In the client URL has been specified and declared asstring.
5. It calls the inter() to connect with the implement() and get theresult.
6. Stop.

# SOURCE CODE

**RMI SERVER:**

import java.rmi.\*; import java.net.\*; public class RmiServer

{

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

{

try

{

FactImplementation fi = new FactImplementation(); Naming.rebind("server", fi); System.out.println("Server reasdy");

}

catch(Exception e)

{

System.out.println("Exception:" + e);

}

}

}

# RMICLIENT:

import java.rmi.\*; importjava.io.\*;

public class RmiClient

{

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

{

try

{

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

System.out.println("Enter the server name or ip address"); String url = br.readLine();

String host="rmi://" + url + "/server";

FactInterface serv = (FactInterface) Naming.lookup(host);

System.out.println("Enter a number:"); int n = Integer.parseInt(br.readLine());

int fact = serv.fact(n);

System.out.println("Factorial of " + n + " is " + fact);

}

catch(Exception e)

{

System.out.println("Error " + e);

}

}

}

# FACT IMPLEMENTATION:

import java.rmi.\*; importjava.rmi.server.\*;

public class FactImplementation extends UnicastRemoteObject implements FactInterface

{

public FactImplementation() throws RemoteException

{

}

public int fact(int n) throws RemoteException

{

int f = 1;

for(int i = 1; i <= n; i++)

{

f = f \* i;

}

return f;

}

}

# FACT INTERFACE:

import java.rmi.\*; importjava.rmi.server.\*;

public interface FactInterface extends Remote

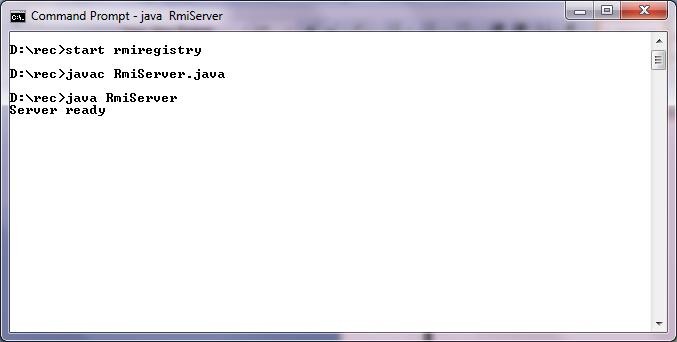
{

public int fact(int n) throws RemoteException;

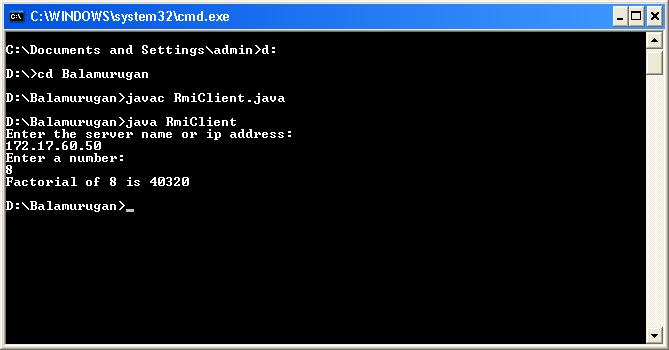
}

# OUTPUT

**RMI SEVRER:**



**RMI CLIENT:**



**RESULT:**

Thus the java program is executed and output is verified successfully.

|  |  |
| --- | --- |
| **Ex. No:7** | **REMOTE METHOD INVOCATIONBASED APPLICATION** |
| **Date:** |

# Aim:

To implement a java for air ticket booking using RMI. **Algorithm:**

# Remote Interface Program

1. Start.
2. Import the java.rmipackage.
3. Declare an interface called “AirTicket”, extending the Remoteinterface.
4. Inside “AirTicket” interface, declare the methods getTicketCost(), getTotalCost(), and getArrivalTime(), each of which throws theRemoteException.
5. Stop.

# Remote Interface Implementation Program

* 1. Start.
  2. Import the java.rmi package and the java.rmi.server.UnicastRemoteObjectclass.
  3. Declare a new class called “AirTicketImpl” extending UnicastRemoteObject and implementing AirTicketinterface.
  4. Define the method getTicketCost() to take in flight-option as an argument and return the cost of aticketcorrespondingtoeachflight,usingswitchcase.Return-1iftheflight-optionisinvalid.
  5. Define the method getTotalCost() to take in the cost of a ticket and no. of tickets reserved as arguments, and return the product of the two arguments as the totalcost.
  6. Define the method getArrivalTime() to take in the flight-option and the departure time as arguments and return the appropriate arrival time corresponding to each flight, using switch case, in 24-hour clock representation. Return -1 if the flight-option isinvalid.
  7. Stop.

# Server Program

1. Start.
2. Import the java.rmipackage.
3. Declare the new class “AirTicketServer”.
4. Inside “AirTicketServer” class, create an object of “AirTicket” interface named “t” by calling the AirTicketImpl() constructor.
5. Bind the remote object to the server using the Naming.rebind() method, passing the string “rmi://localhost:1099/AirTicketDetails” and the interface object “t” asarguments.
6. In the main() method, call the AirTicketServer()constructor.
7. Stop.

# Client Program

1. Start.
2. Import the java.rmi and the java.utilpackages.
3. Declare the new class“AirTicketClient”.
4. In class “AirTicketClient”, declare an object of interface “AirTicket” called“t”.
5. Look up the remote object using the Naming.lookup() method, passing the string “//localhost:1099/AirTicketDetails” as argument, and typecasting and storing the returned object in “t”.
6. Create a new Scanner object that acts as an input stream from theuser.
7. Display the menu of flightoptions.
8. Prompt and read in the flight option chosen from the user using the Scannerobject.
9. Display the menu of flight timesavailable.
10. Prompt and read in the flight time chosen by the user using the Scannerobject.
11. Prompt and read in the no. of tickets to be reserved using the Scannerobject.
12. Invoke the getTicketCost() method, passing the flight-option as parameter, and store the returned result in new variable“ticketCost”.
13. Invoke the getTotalCost() method, passing “ticketCost” and the no. of tickets reserved as arguments, and store the returned result in new variable“totalCost”.
14. Invoke the getArrivalTime() method, passing the flight-option and the flight-time as arguments, and store the returned result in new variable“arrivalTime”.
15. If “ticketCost”, “totalCost”, or “arrivalTime” is less than zero, then display error and exit program.
16. Else, display the Air Ticket Details, including the flight name, source location, destination, departure and arrival times, no. of tickets reserved, cost per ticket, and totalcost.
17. Stop.

# AirTicket.java

import java.rmi.\*;

public interface AirTicket extends Remote

{

public double getTicketCost(int op) throws RemoteException;

public double getTotalCost(double cost, int n) throws RemoteException; public int getArrivalTime(int op, int departTime) throws RemoteException;

}

**AirTicketImpl.jaa** import java.rmi.\*;

importjava.rmi.server.UnicastRemoteObject;

public class AirTicketImpl extends UnicastRemoteObject implements AirTicket { AirTicketImpl() throws RemoteException {}

public double getTicketCost(int op) throws RemoteException{ switch(op){

case1:

return (5000\*1.10); case2:

return (7000\*1.1); case3:

return (4000\*1.10); case4:

return (10000\*1.10); case5:

return (9000\*1.1); default:

return (-1);

}}

public double getTotalCost(double cost, int n) throws RemoteException{ return (cost\*n);

}

public int getArrivalTime(int op, int departTime) throws RemoteException{ int time;

switch(op){ case 1:

time = departTime + 700; break;

case 2:

time = departTime + 1000; break;

case 3:

time = departTime + 500; break;

case 4:

time = departTime + 1400; break;

case 5:

time = departTime + 1200; break;

default: return (-1);

}

if(time >= 2400){

time -= 2400;

}

return time;

}}

**AirTicketServer.java** import java.rmi.\*;

public class AirTicketServer { AirTicketServer() throws Exception

{

AirTicket t = new AirTicketImpl(); Naming.rebind("rmi://localhost:1099/AirTicketDetails", t);

}

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

{

new AirTicketServer();

}

}

# AirTicketClient.java

import java.rmi.\*; importjava.util.Scanner;

public class AirTicketClient {

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

{

AirTicket t = (AirTicket) Naming.lookup("//localhost:1099/AirTicketDetails"); Scanner scan = new Scanner(System.in);

System.out.println("Air Ticket Details Application Using RMI\n\n"); System.out.println("\nFlights:\n1.Chennai -> Frankfurt\n2.Chennai -> London\n3.Chennai -> Dubai\n4.Chennai -> Toronto\n5.Chennai -> Quebec");

System.out.print("Choose a flight: "); int flightop = scan.nextInt();

System.out.println("\nFlight Times:\n 0000 \n 0300 \n 0900 \n 1400\n 1700"); System.out.print("Choose a flight time: ");

int timeop = scan.nextInt();

System.out.println("\nEnter the no. of tickets desired: "); int n = scan.nextInt();

double ticketCost = t.getTicketCost(flightop); double totalCost = t.getTotalCost(ticketCost,n);

int arrivalTime = t.getArrivalTime(flightop, timeop); if(ticketCost<0 || totalCost<0 || arrivalTime<0)

{

System.out.println("Error in processing. Program exiting."); System.exit(0);

}

System.out.println("\n\nYOUR TICKET DETAILS:");

System.out.println("Flight: AirIndia"); System.out.println("Source: Chennai"); System.out.print("Destination: "); switch(flightop)

{

case 1: System.out.println("Frankfurt"); break;

case 2: System.out.println("London"); break;

case 3: System.out.println("Dubai"); break;

case 4: System.out.println("Toronto"); break;

case 5: System.out.println("Quebec"); break;

}

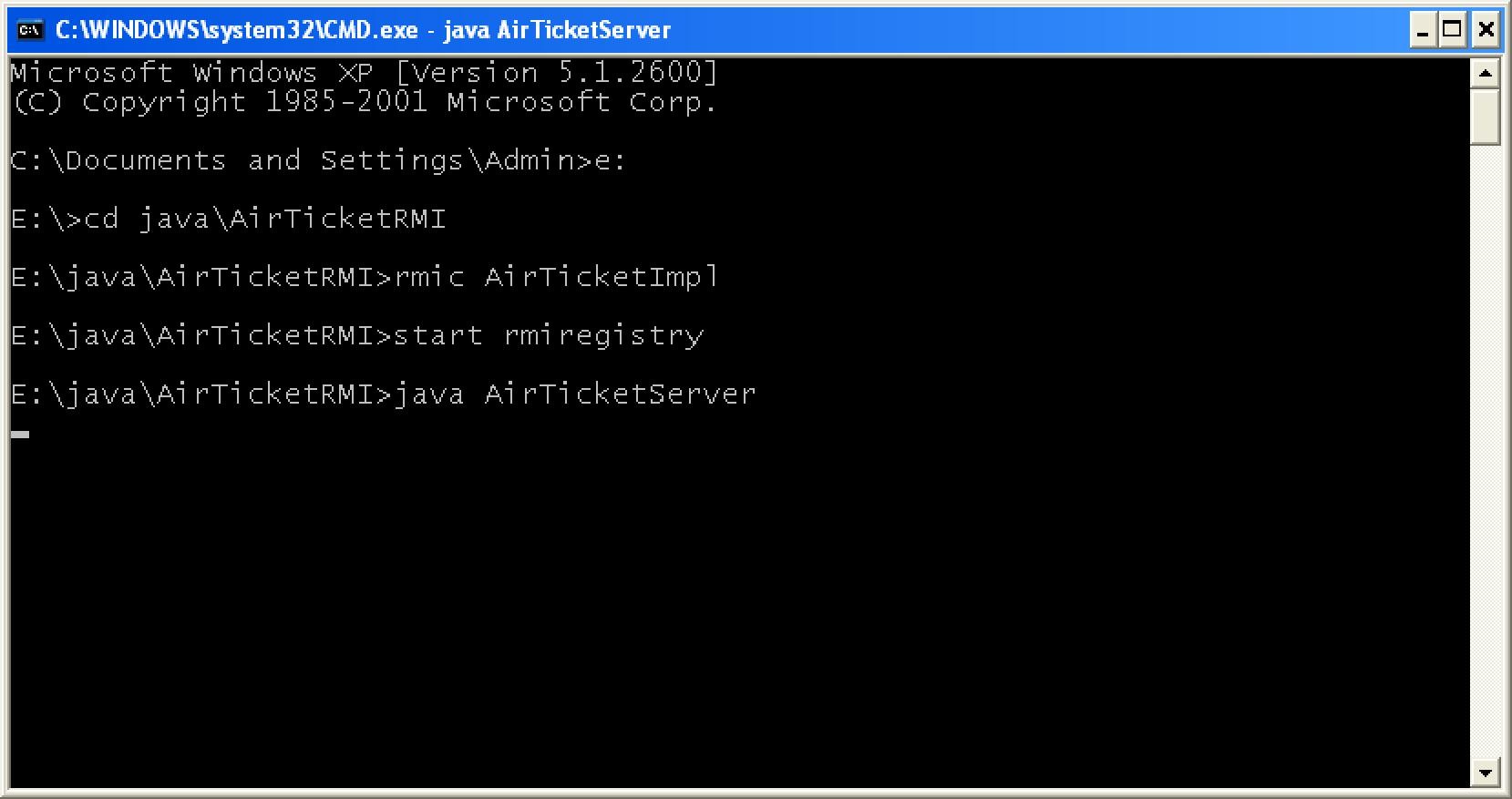
System.out.println("Departure Time: "+timeop); System.out.println("Arrival Time: "+arrivalTime); System.out.println("No. of Tickets Reserved: "+n); System.out.println("Cost per Ticket: "+ticketCost); System.out.println("Total Cost: "+totalCost);

}

}

# Output:

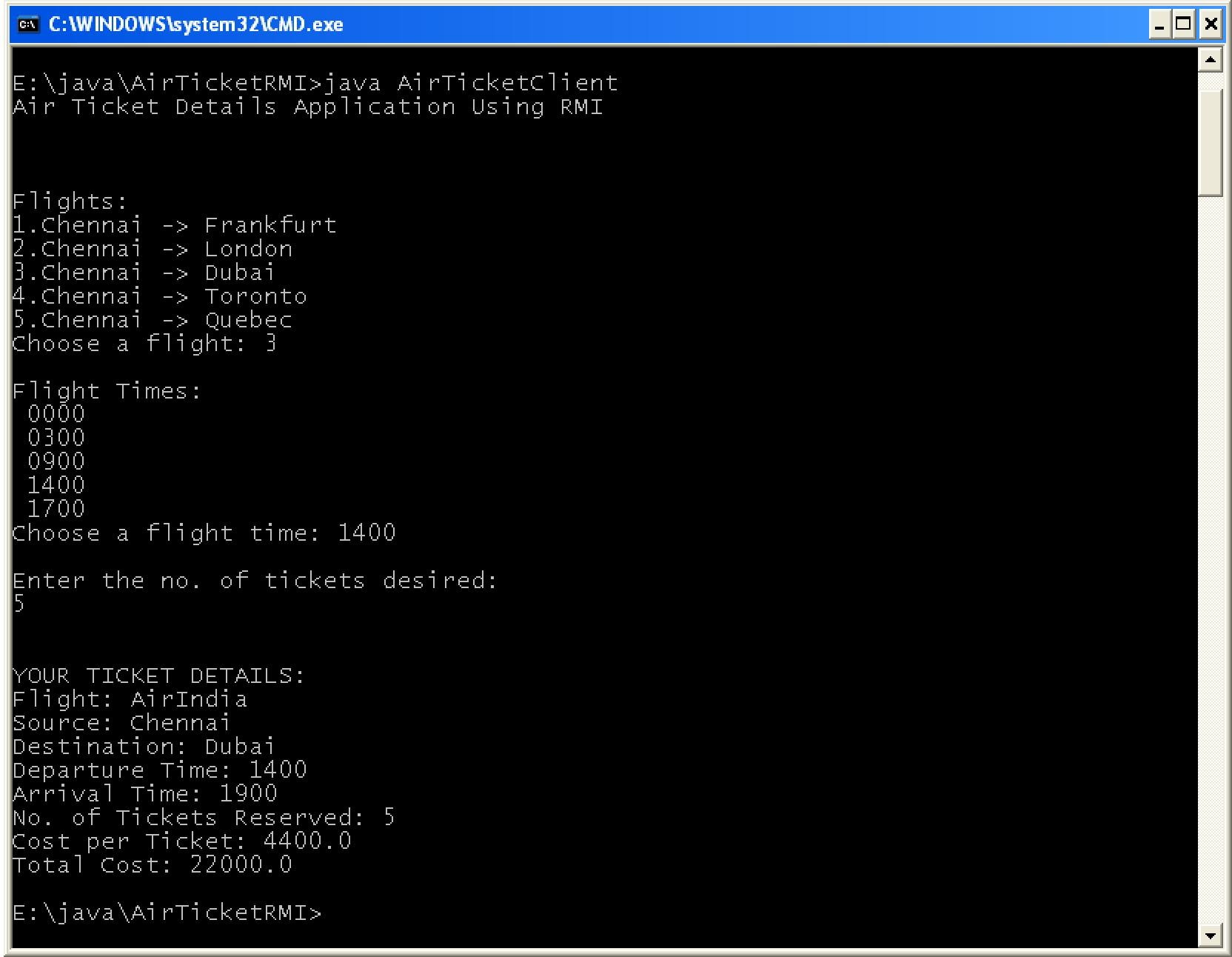
**AirTicketServer.java**



# RMI Registry



**AirTicketClient.java**



**Result:**

Thus program for air ticket booking using remote method invocation is executed and verified.

|  |  |
| --- | --- |
| **Ex. No:8** | **ADDRESS RESOLUTION PROTOCOL** |
| **Date:** |

# Aim:

To implement a java program for finding physical address of a node.

# Algorithm

1. Start.
2. Import the java.netpackage.
3. Declare a new class called“ARP”.
4. Inside class “ARP”, define the main() method to create a new String object called“str”.
5. Create a new object of class InetAddress and initialize it with Local Host IPAddress.
6. Display the InetAddressobject.
7. Convert the InetAddress object to String object and store in“str”.
8. Create a new NetworkInterface object and initialize it using the InetAddressobject.
9. Create a new byte array “MAC” and initialize it using the getHardwareAddress()method.
10. Create a new StringBuilderobject.
11. Repeat the following steps for i = 0 to length of byte array“MAC”:
    1. Assign MAC[i] to String“s”.
    2. Convert value in “s” to Integer type and store in integer type variable“j”.
    3. Convert value in “j” to HexString and store in String“sl”.
    4. If “sl” contains more than 2 characters, then get substring of 2 characters from “sl” and store back in“sl”.
    5. If i ≠ 0, then append “-” before the slvalue.
    6. Append “sl” to the StringBuilderobject.
12. Display the NetworkInterface name using the method getDisplayName() through the NetworkInterface object.
13. Display the MACAddress.
14. Stop

# ARP.java

import java.net.\*; class ARP{

public static void main(String args[]){ InetAddress Ip;

String Str;

System.out.println("\t\tAddress Resolution Protocol"); System.out.println("\t\t----------------------------");

try{

Ip = InetAddress.getLocalHost(); System.out.println("IPAddress\n"+Ip);

Str=Ip.toString(); //Get the MACAddress of the System NetworkInterface Intf = NetworkInterface.getByInetAddress(Ip); byte[] MAC =Intf.getHardwareAddress();

StringBuilder sb = new StringBuilder(); for (int i = 0; i < MAC.length; i++)

{

String s = ""+MAC[i]; if(i < MAC.length - 1){ s = ""+MAC[i];

}

int j=Integer.parseInt(s);

String s1=Integer.toHexString(j); if(s1.length()>2)

{

int l=s1.length()-2; s1 = s1.substring(l);

}

if(i != 0){

s1 = "-"+s1;

}

sb.append(s1);

}

System.out.println(“NetworkInterface Name \n” +Intf.getDisplayName()); System.out.println("\nMACAddress");//Displaying MAC Address at the Client Side for (int i = 0; i < MAC.length; i++)

{

System.out.format("%02X%s", MAC[i], (i < MAC.length - 1) ? "-" : "");

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

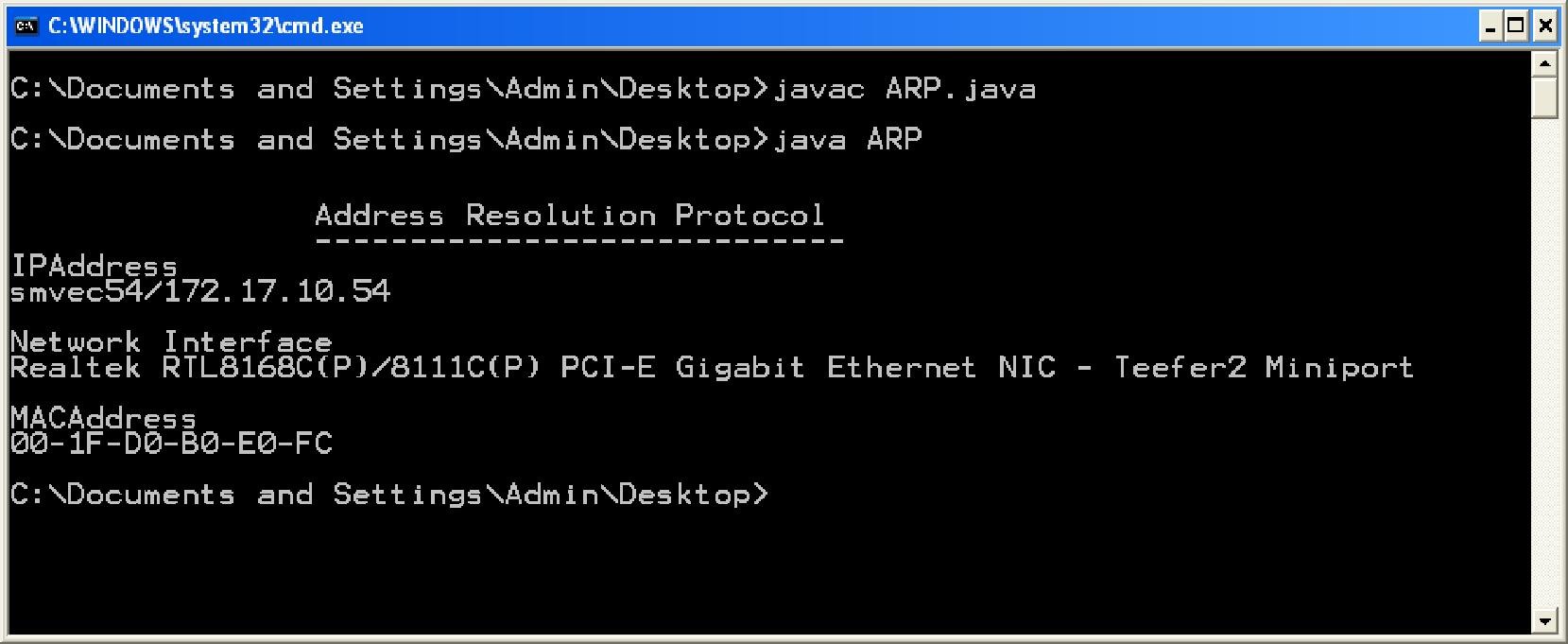
}

System.out.println();

} }

# Output:

**ARP.java**



**Result:**

Thus the program for display physical address of the nodes is executed and verified.

|  |  |
| --- | --- |
| **Ex. No:9** | **CYCLIC REDUNDANCY CHECK** |
| **Date:** |

# Aim:

To implement a java program for Cyclic Redundancy Check.

# Algorithm:

1. Start.
2. Import the java.util and java.iopackages.
3. Declare a new class called “CRC”.
4. Inside class “CRC”, define the main() method to create a new Scanner object to read input from the user.
5. Promptforandreadinthemessagebitsasastringandconvertandstoreinanintegerarray“D”.
6. Prompt for and read in the generator bits as a string and convert and store in an integer array “G”.
7. If G[0] equals 0 then display “Invalid generatorbits”.
8. Declare arrays “DR”, “rem”, and “transmitMessage” of size the sum of array D's and array G's sizes minusone.
9. Copy array D into arrayDR.
10. Display arrays D, G,DR.
11. Copy DR into array rem and call the computeCRC()method.
12. Display the remainder in arrayrem.
13. Ex-or each element of the array rem with array DR and store the elements in array transmitMessage.
14. DisplaytransmitMessage.
15. Prompt for and read in the received message as a String and convert and store as integer in array transmitMessage.
16. Copy transmitMessage into array rem and call the computeCRC()method.
17. If any of the bits in array rem don't equal zero then display “There is error.” Else display “No error.”
18. Stop.

# ComputeCRC()

1. Start.
2. Declare and initialize integer variable “current” tozero.
3. Repeat the followingsteps:
   1. Repeat for i=0 to(g-1):
      1. EX-OR rem[current+i] and G[i] and store back inrem[current+i].
   2. Increment “current” while rem[current] equals zero and while “current” is less than length of array“rem”.
   3. If the no. of elements after “current” position in array “rem” is less than the length of array “G”, then exitloop.
4. Return the array“rem”.
5. Stop.

# Source code:

import java.io.\*; class crc

{

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

{

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

int[] message; int[]gen;

int[] app\_message; int[]rem;

int[] trans\_message;

int message\_bits,gen\_bits, total\_bits; System.out.println("\n Enter number of bits in message : "); message\_bits=Integer.parseInt(br.readLine()); message=new int[message\_bits];

System.out.println("\n Enter message bits : "); for(int i=0; i<message\_bits; i++) message[i]=Integer.parseInt(br.readLine());

System.out.println("\n Enter number of bits in gen : "); gen\_bits=Integer.parseInt(br.readLine());

gen=new int[gen\_bits]; System.out.println("\n Enter gen bits : "); for(int i=0; i<gen\_bits; i++)

{

gen[i]=Integer.parseInt(br.readLine());

}

total\_bits=message\_bits+gen\_bits-1; app\_message=new int[total\_bits]; rem=new int[total\_bits]; trans\_message=new int[total\_bits]; for(int i=0;i<message.length;i++)

{

app\_message[i]=message[i];

}

System.out.print("\n Message bits are : "); for(int i=0; i< message\_bits; i++)

{

System.out.print(message[i]);

}

System.out.print("\n Generators bits are : "); for(int i=0; i< gen\_bits; i++)

{

System.out.print(gen[i]);

}

System.out.print("\n Appended message is : "); for(int i=0; i< app\_message.length; i++)

{

System.out.print(app\_message[i]);

}

for(int j=0; j<app\_message.length; j++)

{

rem[j] = app\_message[j];

}

rem=computecrc(app\_message, gen, rem); for(int i=0;i<app\_message.length;i++)

{

trans\_message[i]=(app\_message[i]^rem[i]);

}

System.out.println("\n Transmitted message from the transmitter is : "); for(int i=0;i<trans\_message.length;i++)

{

System.out.print(trans\_message[i]);

}

System.out.println("\n Enter received message of "+total\_bits+" bits at receiver end : "); for(int i=0; i<trans\_message.length; i++)

{

trans\_message[i]=Integer.parseInt(br.readLine());

}

System.out.println("\n Received message is :"); for(int i=0; i< trans\_message.length; i++)

{

System.out.print(trans\_message[i]);

}

for(int j=0; j<trans\_message.length; j++)

{

rem[j] = trans\_message[j];

}

rem=computecrc(trans\_message, gen, rem); for(int i=0; i< rem.length; i++)

{

if(rem[i]!=0)

{

System.out.println("\n There is Error in the received message!!!"); break;

}

if(i==rem.length-1)

{

System.out.println("\n There is No Error in the received message!!!");

}

}

}

static int[] computecrc(int app\_message[],int gen[], int rem[])

{

int current=0; while(true)

{

for(int i=0;i<gen.length;i++)

{

rem[current+i]=(rem[current+i]^gen[i]);

}

while(rem[current]==0 && current!=rem.length-1)

{

current++;

}

if((rem.length-current)<gen.length)

{

break;

}

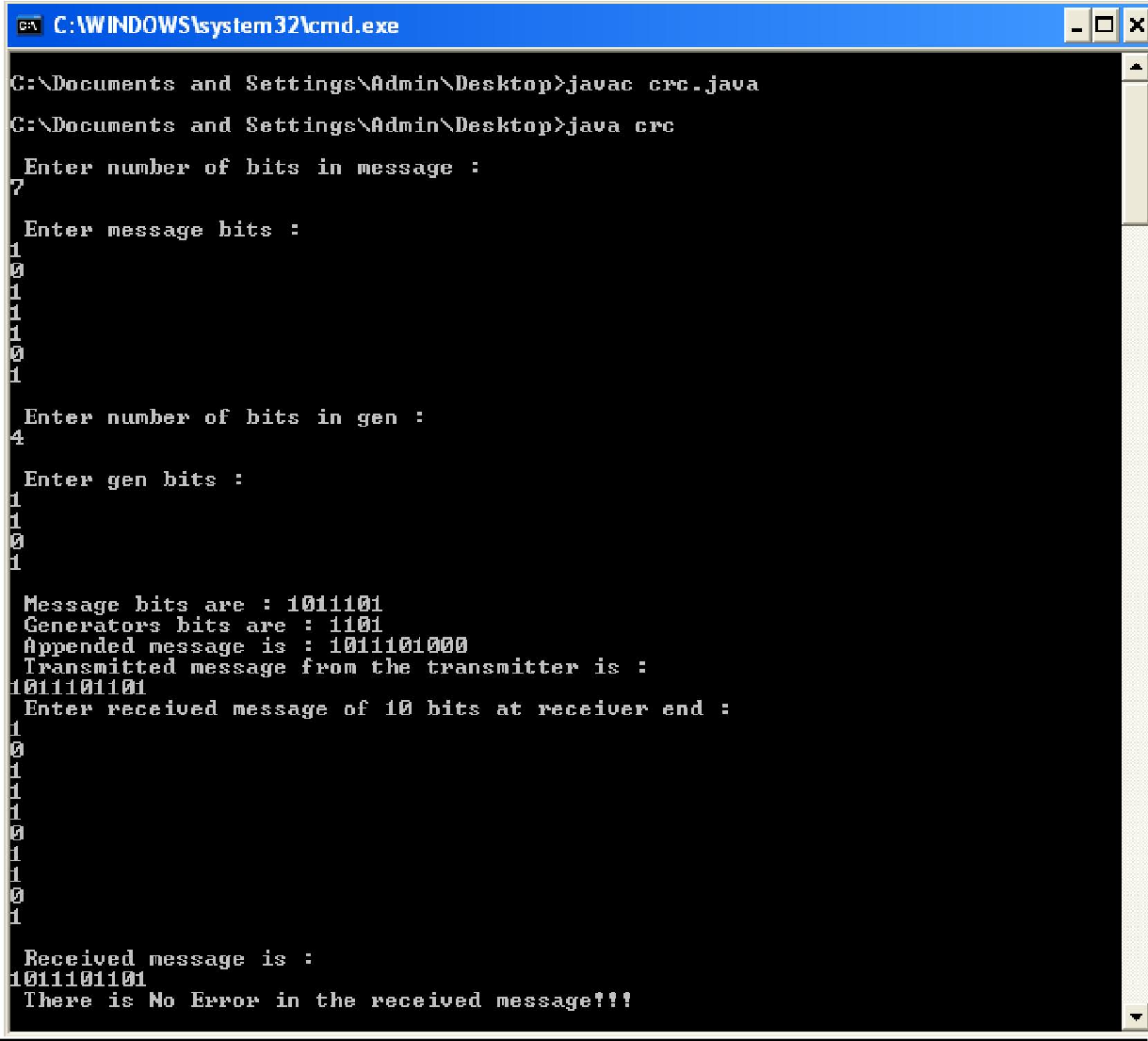
}

return rem;

}

}

# Output:



**Result:**

Thus the program is executed and verified.

|  |  |
| --- | --- |
| **Ex. No: 10** | **SLIDING WINDOW PROTOCOL** |
| **Date:** |

# Aim:

To implement a java program for sliding window protocol.

# Algorithm:

**GBNSend.java**

1. Start.
2. Import the java.net and java.iopackages
3. Declare the class“GBNSend”.
4. Inside class “GBNSend”, declare and initialize the integer variables winStart=0, winEnd, winSize, totalPkt, killPkt, ack=-1, and timeOutEnd=-1, the String array “pkt”, and the boolean array “acknowledged” which is initialized tofalse.
5. Create a BufferedReader object to read in input from theuser.
6. Prompt for and read in the message to be sent, splitting the input string into packets and storing in array“pkt”.
7. Assign the total no. of elements found in array “pkt” into integertotalPkt.
8. Prompt for and read in the window size, storing in integerwinSize.
9. Assign winSize towinEnd.
10. Prompt for and read in the packet no. with which to display packet loss, storing it inkillPkt.
11. Create a new ServerSocket object with port number 6322 and a new Socket object using the accept() method.
12. Create a new BufferedReader object to read in input from the clientsocket.
13. Create a new PrintStream object to send messages to the clientsocket.
14. CreateanewGBNSenderReceivingobjectandstartthethreadtobeginreceivingacknowledgement.
15. Display the total no. of packets to be sent and send this value to theclient.
16. Declare and initialize the boolean variables firstKill = true, and first =true.
17. Repeat the followingsteps:
    1. For each packet “i” in the window of size winSize, and also while acknowledged[i] is not equal to true,do:
       1. if i = killPkt and firstKill = true, then display that packet with the message “Error: Packet Loss” and go to step(i).
       2. if the boolean variable causeSleep = true, then assign false to causeSleep and make the thread sleep for 4seconds.
       3. Make the thread sleep for 1second.
       4. If acknowledged[i] is not equal to true, then display the packet to be sent and send the packet no. and packet data to the client.
       5. Else, repeat from step(i).
    2. If i = totalPkt and ack = totalPkt, then display the no. of packets for which acknowledgement is received totally for and exit theprogram.
    3. Make the thread sleep for 3seconds.
    4. If ack = winStart and first = true, then Display “Start Timeout....Resending Lost packets”, and assign timeOutEnd = winStart and assign first =false.

18. Stop.

# GBNSenderReceiving

1. Start.
2. Declare a class called“GBNSenderReceiving”.
3. In “GBNSenderReceiving”, declare & initialize boolean variable “causeSleep” asfalse.
4. Create a new Thread “t” using “this” reference and start thethread.
5. While ack < totalPkt, do:

i.) Read in acknowledgement no. from the client; decrement and store it in integer “a”. ii.) if a = timeOutEnd, then set causeSleep to true.

iii.)Assign acknowledged[a] = true.

iv.) Display that acknowledgement is received for packet a. v.) if a = timeOutEnd, then display “End Timeout”.

vi.) Increment winStart and winEnd, and assign ack = a+1.

1. Stop.

# GBNRecv.java

1. Start.
2. Import the java.io and java.netpackages.
3. Declare a class called“GBNRecv”.
4. Inside class “GBNRecv”, declare and initialize the integer variables totalPkt, prevAck=0, ack=1, andtimeOutEnd=-1.
5. Create a new Socket object using Local Host IP Address and the port number6322.
6. Create a new BufferedReader object to read in input from theserver.
7. Create a new PrintStream object to send output to theserver.
8. Read in the input from the server and assign it tototalPkt.
9. Declare and initialize the boolean variable “flag” as false, the String “pkt”, the String array “bufData”, the integer array “bufAck”, and the integer b =0.
10. While ack<totalPkt,do:
    1. Read in the packet no. from server and store in integer“n”.
    2. Read in the packet data from server and store in string“pkt”.
    3. If prevAck = n, thendo:
       1. display that packet “n” has been received and acknowledgement issent.
       2. If prevAck = timeOutEnd, then display “End Timeout” and display that acknowledgement is sent for all the packets stored in array bufData before sending all the packet numbers stored in array bufAck to theserver.
       3. Assign prevAck as last packet no. stored in bufAck and assign ack = prevAck +1.
       4. Increment ack and assign prevAck = ack –1.
    4. Else,do:
       1. if flag = false, then assign timeOutEnd = prevAck, assign flag = true, and display “Start Timeout”.
       2. Display that packet “n” is received but no acknowledgement issent.
       3. Insert “n” and packet n's data into arrays bufAck and bufDatarespectively.
11. Display the total no. of packetsreceived.
12. Stop.

# GBNSend.java

import java.io.\*; import java.net.\*; class GBNSend{

static int winStart, winEnd, winSize, totalPkt, killPkt, ack, timeOutEnd; static String pkt[];

static boolean acknowledged[]; static BufferedReader in=null; GBNSend() throws Exception{

System.out.println("Enter the message to be sent:");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in)); pkt = (br.readLine()).split(" ");

acknowledged= new boolean[pkt.length]; for(int i=0; i<pkt.length; i++) acknowledged[i] = false;

totalPkt = pkt.length; System.out.print("\nEnter the window size: "); winSize = Integer.parseInt(br.readLine()); winStart =0;

winEnd = winSize;

ack = timeOutEnd = -1;

System.out.print("\nEnter the packet no. which to be displayed as lost packet(Start from 0): "); killPkt = Integer.parseInt(br.readLine());

}

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

System.out.println("\t\tGoBackN Protocol using Sliding Window Protocol\nSender\n---------"); ServerSocket ss = new ServerSocket(6322);

Socket s=ss.accept();

GBNSend g = new GBNSend();

in = new BufferedReader(new InputStreamReader(s.getInputStream())); PrintStream out = new PrintStream(s.getOutputStream()); GBNSenderReceiving threadReceiving = new GBNSenderReceiving(); System.out.println("\nTotal no. of packets to be sent: "+totalPkt); out.println(totalPkt);

boolean firstKill=true; boolean first=true;

int i; while(true){

for(i=winStart; i>=winStart && i<winEnd && i<totalPkt; i++){ if(acknowledged[i]!=true){

if(i==(killPkt) && firstKill == true){

System.out.println("\nSending packet " + i + ": " + pkt[i]+ "\t-----ERROR: PACKET LOSS."); firstKill = false;

continue;

}

if (GBNSenderReceiving.causeSleep==true){ GBNSenderReceiving.causeSleep=false; Thread.sleep(4000);

}

Thread.sleep(1000); if(acknowledged[i] != true){

System.out.println("\nSending packet " + i + ": " + pkt[i]); out.println(i);

out.println(pkt[i]);

}

else continue;

}}

if(i == totalPkt && ack == totalPkt){

System.out.println("\nAcknowledgement received totally for "+ack+ " Packets!"); in.close();

out.close();

s.close();

ss.close(); System.exit(0);

}

Thread.sleep(3000);

if(ack == winStart && first==true){

System.out.println("\n\t-----START TIMEOUT-----\nResending lost packets......."); timeOutEnd = winStart;

first=false;

}}}}

class GBNSenderReceiving implements Runnable{ inta;

static volatile boolean causeSleep=false; GBNSenderReceiving(){

Thread t = new Thread(this); t.start();

}

public void run(){ try{

while(GBNSend.ack<GBNSend.totalPkt){

a = Integer.parseInt(GBNSend.in.readLine()) - 1; if(a==GBNSender2.timeOutEnd){ causeSleep=true;

}

GBNSend.acknowledged[a] = true;

System.out.println("\nReceived Acknowledgement for packet: " + a); if(a == GBNSend.timeOutEnd){

System.out.println("\n\t-----END TIMEOUT-----");

}

GBNSend.winStart++; GBNSend.winEnd++; GBNSend.ack = a+1;

}}

catch(Exception e){ System.out.println(e)

}}}

**GBNRecv.java** import java.io.\*; import java.net.\*; class GBNRecv{

int totalPkt, prevAck, ack, timeOutEnd;

public static void main(String args[]) throws Exception{ GBNRecv g = new GBNRecv();

System.out.println("\t\tGoBackN Protocol using Sliding Window Protocol\nReceiver\n---------"); Socket s = new Socket("LocalHost", 6322);

BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream())); PrintStream out = new PrintStream(s.getOutputStream());

g.totalPkt = Integer.parseInt(in.readLine()); g.timeOutEnd=-1;

* 1. revAck=0; g.ack=1;

boolean flag = false; String pkt;

int n;

int bufAck[] = new int[g.totalPkt];

String bufData[] = new String[g.totalPkt]; int b=0;

while(g.prevAck<g.totalPkt){

n = Integer.parseInt(in.readLine()); pkt = in.readLine();

if(g.prevAck == n){

System.out.println("\nPacket " + g.prevAck + " Received:" + pkt + "\t Acknowledgement sent!"); out.println(g.ack);

if(g.prevAck==g.timeOutEnd){ System.out.println("\n\t-----END TIMEOUT-----"); for(int j=0; j<b; j++){

System.out.println("\nAcknowledgement sent for Packet "+ bufAck[j] + ": " +bufData[j]); out.println(bufAck[j]+1);

}

g.prevAck = bufAck[b-1]; g.ack = g.prevAck +1;

}

g.ack ++;

g.prevAck = g.ack-1;

}

else{ if(flag==false){

g.timeOutEnd = g.prevAck;

System.out.println("\n\t-----START TIMEOUT-----"); flag=true;

}

System.out.println("\nPacket " + n + " Received: " + pkt + "\t NO Acknowledgement sent!"); bufAck[b] = n;

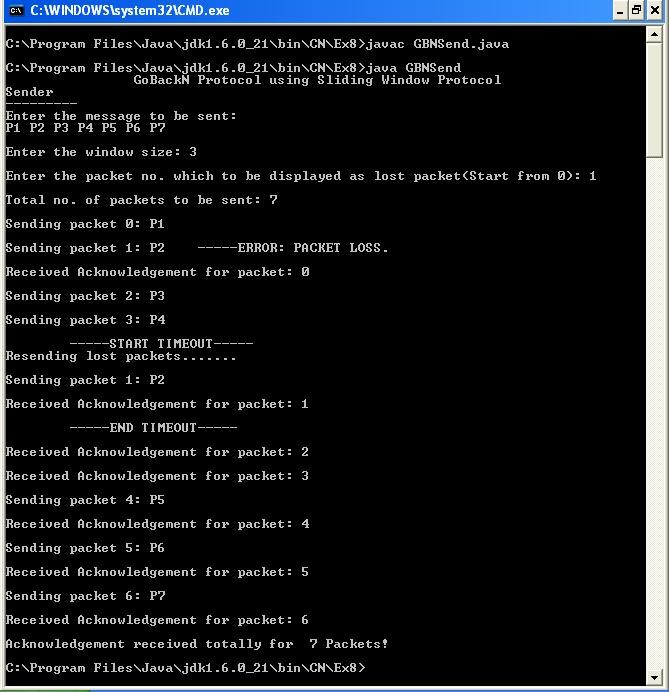
bufData[b] = pkt; b= b+1;

}}

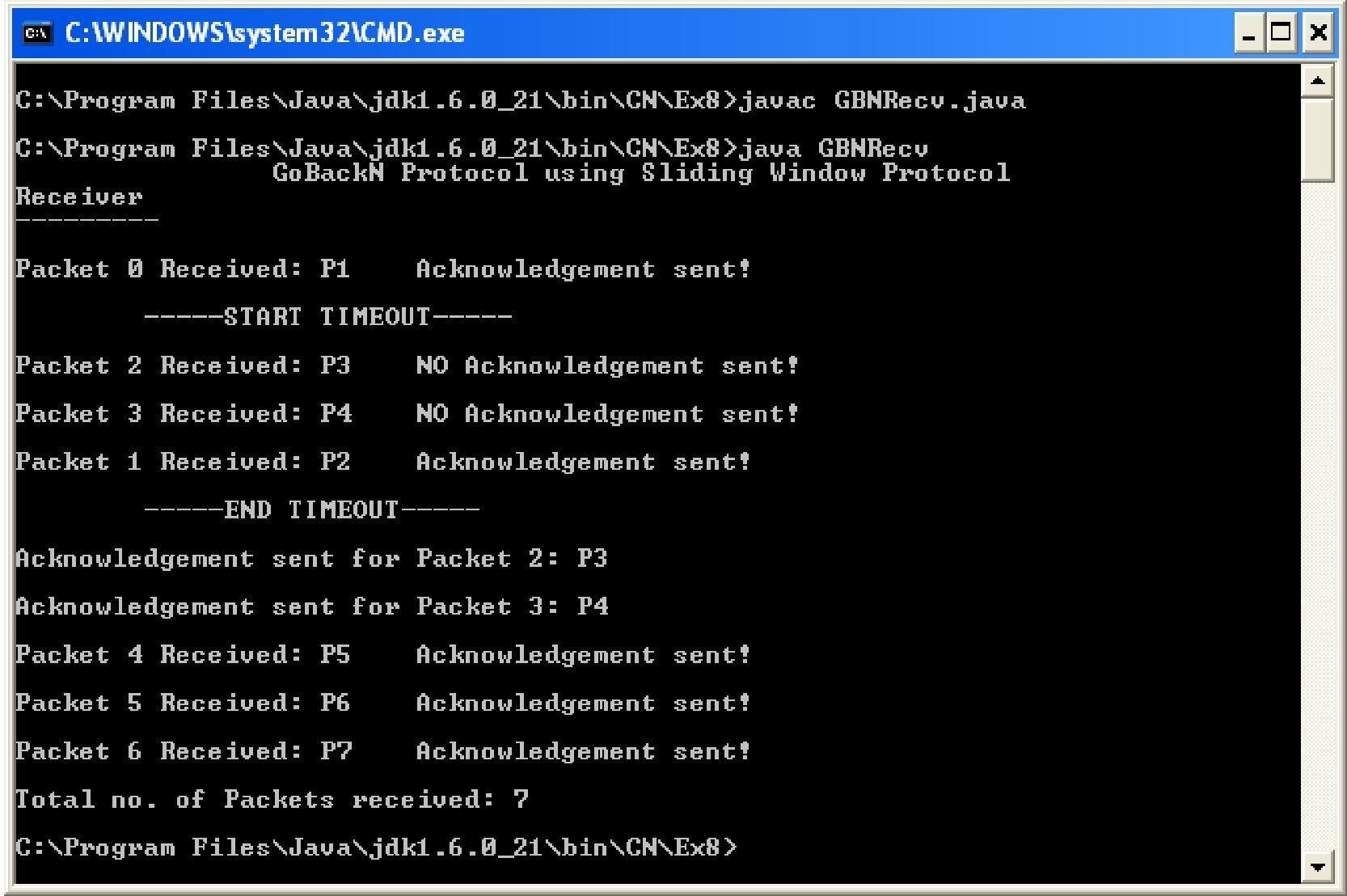
System.out.println("\nTotal no. of Packets received: "+g.prevAck);

}}

# Output: GBNSend.java



**GBNRecv.java**



**Result:**

Thus the program is executed and verified.

|  |  |
| --- | --- |
| **Ex. No:11** | **HTTP WEBPAGE DOWNLOAD (RESPONSE CODE)** |
| **Date:** |

# Aim:

To write a java program for HTTP webpage download.

# Algorithm:

* + 1. Start.
    2. Import the java.net and java.iopackages.
    3. Declare the class“HttpResp”.
    4. Inside class “HttpResp”, read in the command linearguments.
    5. For each command line argument given as input,do:
       1. Create a new URL object using the command lineargument.
       2. Create a new HttpURLConnection object using the openConnection() method and typecasting it as HttpURLConnection.
       3. Create a new URLConnection object using the openConnection()method.
       4. Display “HTTP ResponseMessage”.
       5. Display the URLobject.
       6. Display the host name of the URL using the getHost()method.
       7. Display the Protocol of the URL using the getProtocol()method.
       8. Display the path of the URL using the getPath() method.
       9. Display the query of the URL using the getQuery() method.
       10. Display the file name of the URL using the getFile()method.
       11. Retrieve and store the connection response code in an integer variable “code”, using the getResponseCode() method.
       12. Retrieve and store the connection response message in a String object, using the getResponseMessage()method.
       13. Display the connection response code andmessage.
       14. Repeat the following steps, initializing integer j = 0:
           1. Get the header field key and the header field value using the methods getHeaderFieldKey(j) and getHeaderField(j)respectively.
           2. If the header field key or the header field values are null then exit theloop.
           3. Display the header field key and the header fieldvalue.
       15. Create a new BufferedReader object to read in the input from theURL.
       16. Declare and initialize integer variable c =1.
       17. Read in each line from the URL using the BufferedReader object and while the input read is not null and c is less than or equal to 10,do:
           1. Display theline.
           2. Increment c byone.
    6. Stop.

# HttpResp.java

import java.net.\*; import java.io.\*; public classHttpResp{

public static void main ( String[] args ) throws IOException{ for(int i=0;i<args.length;i++){

try{

URL url = new URL(args[i]);

HttpURLConnection connection = (HttpURLConnection)url.openConnection(); URLConnection conn = url.openConnection(); System.out.println("\n\t\t\tHTTP RESPONSE MESSAGE\n"); System.out.println("URL:"+url+"\n");

System.out.println("HOST NAME : "+url.getHost()); System.out.println("PROTOCOL : " + url.getProtocol()); System.out.println("PATH : " + url.getPath()); System.out.println("QUERY : " + url.getQuery()); System.out.println("FILE NAME : " + url.getFile());

int code=connection.getResponseCode();

String response=connection.getResponseMessage(); System.out.println("STATUS : "+connection.getResponseMessage()); System.out.println("RESPONSE CODE : "+ connection.getResponseCode()); for (int j=1;; j++)

{

String name = connection.getHeaderFieldKey(j); String value = connection.getHeaderField(j);

if (name == null || value == null) break;

System.out.println(connection.getHeaderFieldKey(j)+":"+value+"\n");

}

System.out.println("\t\t=== Content ===");

BufferedReader br = new BufferedReader(new InputStreamReader(connection.getInputStream())); int c=1;

String l; while(((l=br.readLine())!=null)&&(c<=10))

{

System.out.print(l); c++;

}

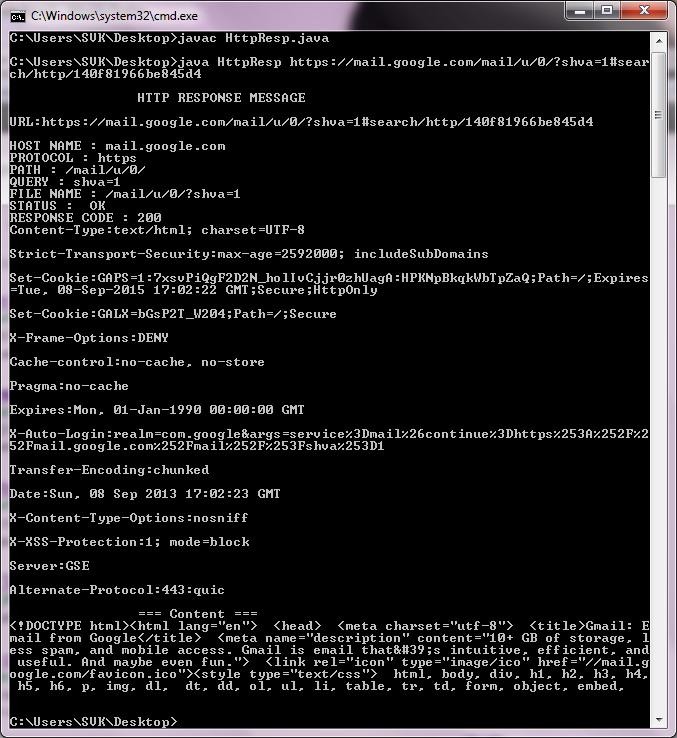
System.out.println("\n");

}

catch (Exception e) {}

}}}

# Output: HttpResp.java



**Result:**

Thus the program is executed and verified.

|  |  |
| --- | --- |
| **Ex. No:11.1** | **SHORTEST PATH ROUTING** |
| **Date:** |

# Aim:

To write a java program for shortest path routing.

# Algorithm:

1. Start.
2. Import the java.net and java.iopackages.
3. Declare a class called“Shpath”.
4. Inside class “Shpath”, declare the integer variables n,s,d,i,j,y=0,sd=9999,al=1, and the integer arrays in, dis,path,m.
5. Create a new BufferedReader object to get userinput.
6. Prompt for and read in the no. of nodes, the source node, the destination node, and the edge distances present between thenodes.
7. Increment al by one and assign al toj.
8. Repeat for i= 1 ton:
   1. Assign in[i]=0;
   2. Assign m[s][i] todis[i].
   3. If m[s][i] does not equal zero then assign path[i] =s;
9. Assign in[s]=1 anddis[s]=0;
10. Repeat for i=2 ton-1:
    1. Repeat for j=1 ton:
       1. If in[j]=0 then
          1. If dis[j] is less than or equal to sd then assign sd=dis[j] andy=j.
       2. Assign in[y]=1.
11. Repeat for j=1 ton:
    1. If in[j]=0 and m[y][j] does not equal zero,then
       1. If sum of dis[y] and m[y][j] is less than dis[j],them
          1. Assign dis[j] = dis[y] +m[y][j]
          2. Assign path[j] =y;
12. Assign i=d;
13. Display “Shortest path is” and display the value ofs.
14. Repeat while path[i] does not equals:
    1. Display “” and the value inpath[i].
    2. Assign i =path[i].
15. Display “” and the value ofd.
16. Display “Distance of Shortest Path is” and append the value indis[d].
17. Stop.

# Shpath.java

import java.net.\*; import java.io.\*; class Shpath{

public static void main(String args[]) throws IOException

{

int n,s,d,i,j,y=0,sd=9999; int[] in=new int[10];

int a1=1;

int[] dis=new int[10]; int[] path=new int[10];

BufferedReader a=new BufferedReader(new InputStreamReader(System.in)); System.out.println("\n\t\tSHORTEST PATH ROUTING"); System.out.print("\nEnter the number of Nodes:"); n=Integer.parseInt(a.readLine());

System.out.print("\nEnter the Source Node:\t"); s=Integer.parseInt(a.readLine()); System.out.print("\nEnter the Destination Node:"); d=Integer.parseInt(a.readLine());

int[][] m=new int[n+1][n+1]; j=a1;

for(i=1;i<n;i++){ while(j<n){

System.out.print("\nEnter distance between [" +i+ "," +(j+1)+"]:"); m[i][j+1]=Integer.parseInt(a.readLine());

m[j+1][i]=m[i][j+1]; j++;

} j=a1+1; a1=a1+1;

}

for(i=1;i<=n;i++){ in[i]=0; dis[i]=m[s][i];

if(m[s][i]!=0){ path[i]=s;

}

}

in[s]=1;

dis[s]=0; for(i=2;i<n;i++){ for(j=1;j<=n;j++){ if(in[j]==0){

if(dis[j]<=sd){ sd=dis[j];

y=j;

}

}

}

in[y]=1; for(j=1;j<=n;j++)

{

if((in[j]==0)&&(m[y][j]!=0))

{

if((dis[y]+m[y][j])<dis[j])

{

dis[j]=dis[y]+m[y][j]; path[j]=y;

}

}

}

}

i=d; System.out.println("\n");

System.out.println("\n The Shortest Path is : \n"); System.out.print(" "+s);

while(path[i]!=s)

{

System.out.print("----> " +path[i]); i=path[i];

}

System.out.print("---->"); System.out.println(d); System.out.print("\n");

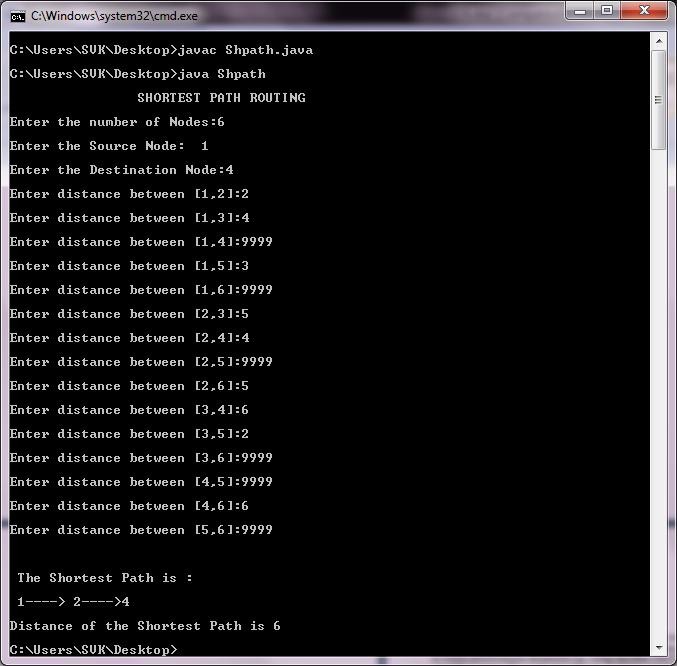
System.out.println("Distance of the Shortest Path is "+dis[d]);

}

}

# OUTPUT

# Shpath.java



**Result:**

Thus the program is executed and verified.

|  |  |
| --- | --- |
| **Ex. No: 12.1** | **DISTANCE VECTOR ROUTING** |
| **Date:** |

# Aim:

To write a java program for distance vector routing.

# Algorithm:

1. Start.
2. Import the BufferedInputStream and the BufferedReader classes from the java.IOpackage.
3. Declare the class“DV”.
4. Inside class “DV”, create the integer arrays “dist” and“from”.
5. Define the main() method to create a new BufferedReader object to get input fromuser.
6. Create an array of DV objects called“rt”.
7. Declare the integer variables i,j,k,count=0, r,m,c, and the two-dimensional array“costmat”.
8. Prompt for and read in the no. ofnodes.
9. For each node “i”do:
   1. Prompt for and read in the distance from node “i" to each of the other nodes “j” and store in rt[i].dist[j] andcostmat[i][j].
   2. Store the value of j inrt[i].from[j].
10. Assign count=0.
11. For each node “i”do:
    1. For each node “j”do:
       1. for each node “k”do:
          1. if rt[i].dist[j]>costmat[i][k]+rt[k].dist[j], then assign rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j] and assignrt[i].from[j]=k;
          2. Incrementcount.
12. Repeat from step 10 while count doesn't equalzero.
13. For each node “i”do:
    1. Display “Shortest distance for” router“i”.
    2. For each node “j”do:
       1. Display node “j”, through which node to “j”, and the distance from node “i” to “j”.
14. Prompt for and read in the required routerno.
15. For each node “m”, display the destination (m), the hop (rt[r-1].from[m]), and the cost (rt[r- 1].dist[m]).
16. Stop.

# DV.java

import java.io.\*;

import java.io.BufferedInputStream; import java.io.BufferedReader; classDV

{

public int dist[]=new int[20]; public int from[]=new int [20];

public static void main(String args[]) throws IOException

{

BufferedReader a=new BufferedReader(new InputStreamReader(System.in)); DV rt[]=newDV[10];

for(intn=0;n<10;n++)

{

rt[n]=new DV();

}

int costmat[][]=new int[20][20]; int nodes,i,j,k,count=0,r,m,c;

System.out.println("\n\t\t DISTANCE VECTOR ROUTING "); System.out.print("Enter the number of nodes : "); nodes=Integer.parseInt(a.readLine());//Enter the nodes for(i=0;i<nodes;i++)

{

System.out.println("Enter the cost matrix for router "+(i+1)); for(j=0;j<nodes;j++)

{

costmat[i][j]=Integer.parseInt(a.readLine()); costmat[i][i]=0;

rt[i].dist[j]=costmat[i][j];//initialise the distance equal to cost matrix rt[i].from[j]=j;

}

}

do

{

count=0;

for(i=0;i<nodes;i++)//We choose arbitary vertex k and we calculate the direct distance from the node i to k using the costmatrix

{//and add the distance from k to node j for(j=0;j<nodes;j++)

{

for(k=0;k<nodes;k++)

{

if(rt[i].dist[j]>costmat[i][k]+rt[k].dist[j])

{//We calculate the minimum distance rt[i].dist[j]=rt[i].dist[k]+rt[k].dist[j]; rt[i].from[j]=k;

count++;

}

}

}

}

}while(count!=0); for(i=0;i<nodes;i++)

{

System.out.println("\nShortest Distance for router "+(i+1)); for(j=0;j<nodes;j++)

{

System.out.println("\tNode "+(j+1)+" via "+(rt[i].from[j]+1)+" Distance "+(rt[i].dist[j]));

}

}

System.out.print("\n Enter the Required router number:"); r=Integer.parseInt(a.readLine());

System.out.println("\n DESTINATION \t HOP \t COST \n"); for(m=0;m<nodes;m++)

{

System.out.println(""+(m+1)+" \t\t "+(rt[r-1].from[m]+1)+"\t "+rt[r-1].dist[m]);

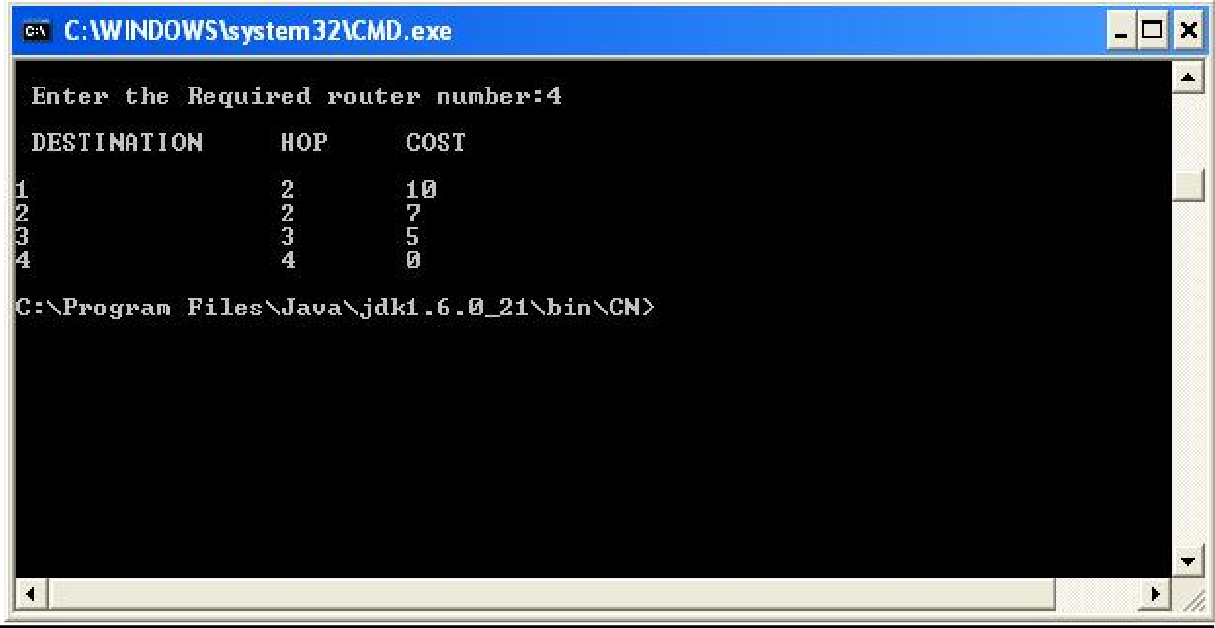
}

}

}

# Output: DV.java

**DV.java**



**Result:**

Thus the program is executed and verified.

|  |  |
| --- | --- |
| **Ex. No:11.2** | **MULTICAST ROUTING** |
| **Date:** |

# Aim:

To write a java program for multicast routing

# Algorithm:

1. Start.
2. Import the java.io and java.netpackages.
3. Declare the class“Multicast”.
4. Inside class “Multicast”, declare the String objects ch, ip,rd.
5. Create a new DataInputStream object to read input from theuser.
6. Prompt for and read in a Class D IP Address (Range: 224.0.0.0 –239.255.255.255).
7. Create an InetAddress object using the getByName() method and the Class D IPAddress.
8. Display the InetAddressobject.
9. Create a new MulticastSocket object with port number 6899 and join the Class D IP Address group using the joinGroup()method.
10. Prompt for and read in if a message is to be sent. If “yes”, go to next step. Else,go to step 16.
11. Prompt for and read in themessage.
12. Encapsulate the message in a DatagramPacket object and send it to the group IPAddress.
13. Create a new DatagramPacket object to receive themessage.
14. Get the data from the DatagramPacket object and displayit.
15. Go to step 10.
16. Display “You're in the listeningmode!”
17. Create a new DatagramPacket object to receive themessage.
18. Get the data from the DatagramPacket object and displayit.
19. Prompt for and read in the user's choice of continuing in listening mode. If “yes”, go to step 16. Else, go to step10.
20. Stop.

# Multicast.java

import java.net.\*; import java.io.\*; public class Multicast

{

public static void main (String args[]) throws IOException

{

String ch,ip,rd; System.out.println("\n\t\tMULTICASTROUTING");

DataInputStream dis=new DataInputStream(System.in);

System.out.println("Enter Class D IP(Range:224.0.0.0 -239.255.255.255) to Form/Join group "); ip=dis.readLine();

InetAddress group = InetAddress.getByName(ip); System.out.print(group);

MulticastSocket s = new MulticastSocket(6899); s.joinGroup(group);

System.out.print("\nWelcome to the group "+ip+"!!"); do

{

do

{

System.out.print("\nWanna send message to a group (Y/N)?"); ch=dis.readLine();

if(ch.equalsIgnoreCase("Y"))

{

System.out.print("Enter the message :"); String msg=dis.readLine();

DatagramPacket hi = new DatagramPacket(msg.getBytes(), msg.length(),group, 6899); s.send(hi);

byte[] buf = new byte[50];

DatagramPacket recv = new DatagramPacket(buf, buf.length); s.receive(recv);

int m= (recv.getLength());

ch= new String(recv.getData()); StringBuffer l=new StringBuffer(ch); l.setLength(m);

System.out.print("\nThe Message send to the group is "+l); System.out.print("\nWanna send more Message (Y/N)?"); ch=dis.readLine();

}

else break;

}while(ch.equalsIgnoreCase("Y")); do

{

System.out.println("\nYou are in the listening mode!!"); byte[] buf = new byte[50];

DatagramPacket recv = new DatagramPacket(buf, buf.length); s.receive(recv);

int d= (recv.getLength());

ch= new String(recv.getData()); StringBuffer f=new StringBuffer(ch); f.setLength(d);

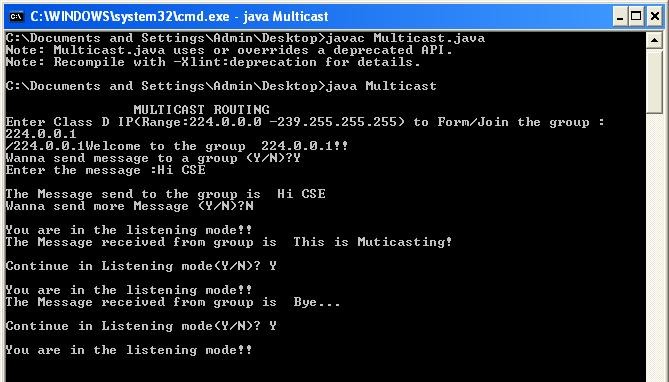
System.out.println("The Message received from group is "+f); System.out.print("\nContinue in Listening mode(Y/N)? "); rd=dis.readLine();

}while(rd.equalsIgnoreCase("Y"));

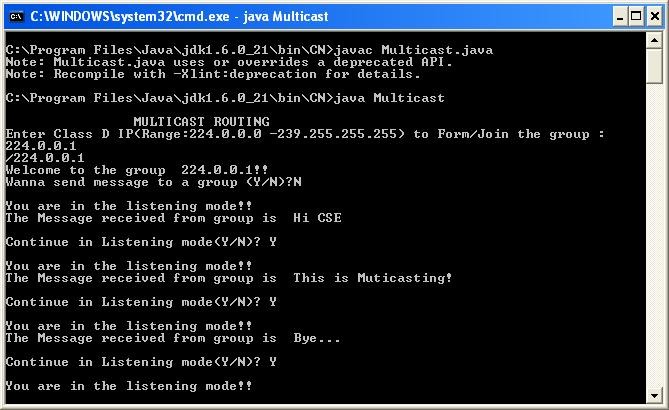
}while(true);

}}

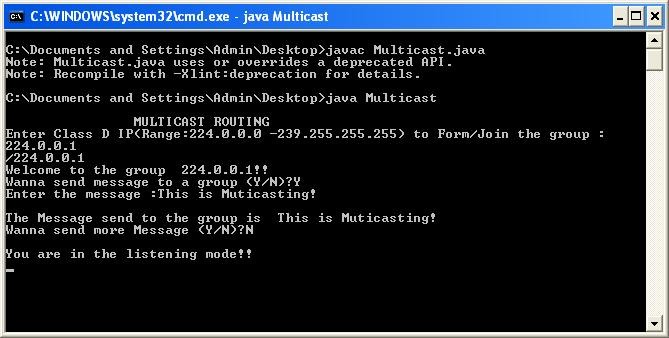
# Output: SENDER



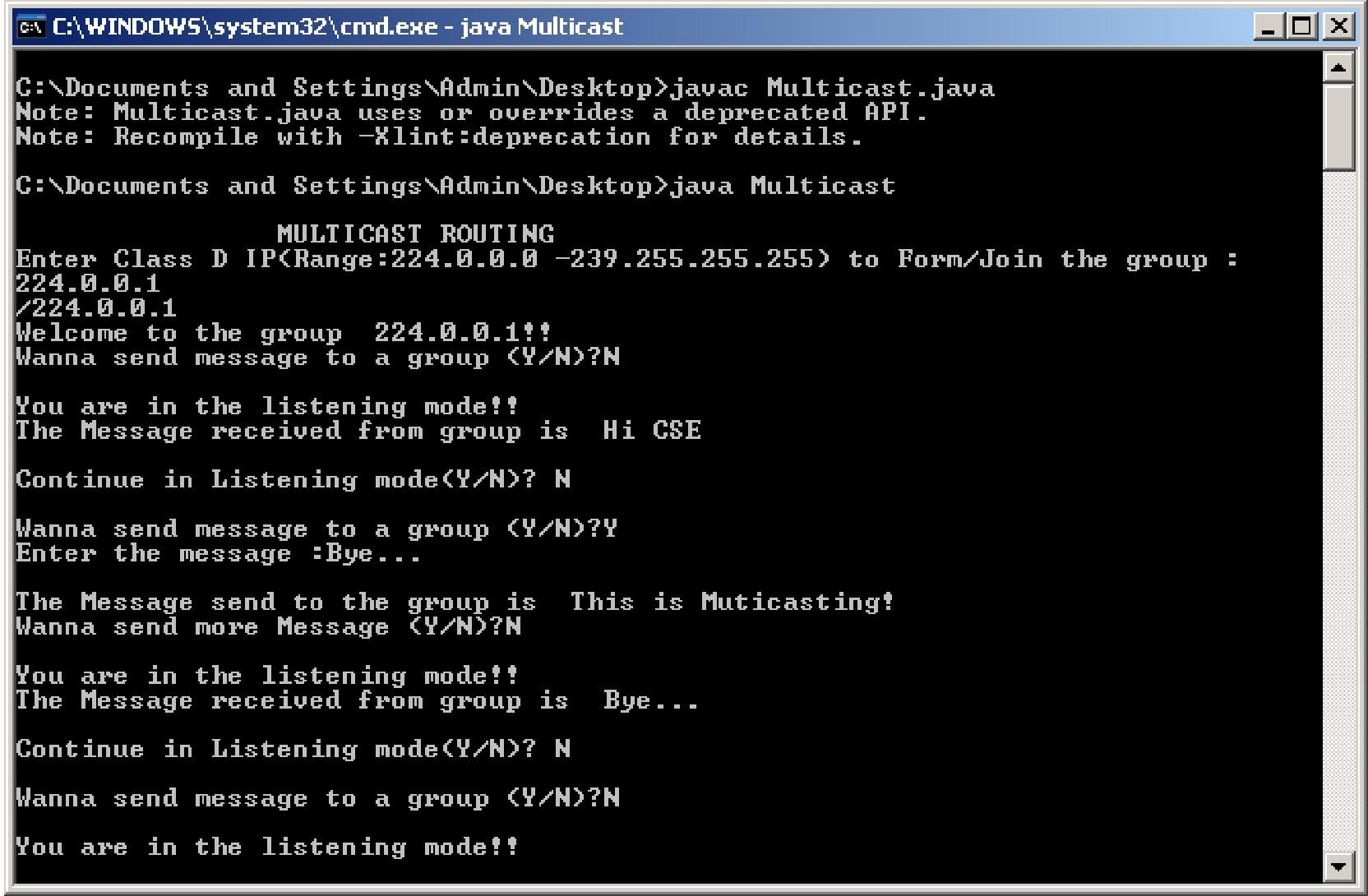
**RECEIVER1**



**RECEIVER2**



**RECEIVER3**



**Result:**

Thus the program is executed and verified.

|  |  |
| --- | --- |
| **Ex. No:12** | **IMPLEMENTATION USING NS2** |
| **Date:** |

# Aim:

To study the implementation of routing using NS2

# Introduction:

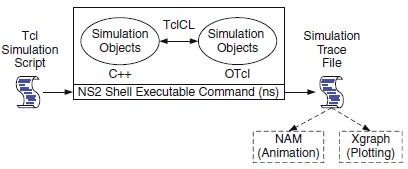
Network Simulator (Version 2),is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989.Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field .Among these are the University of California and Cornell University who developed the REAL network simulator,1 the foundation which NS is based on. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual Inter Network Testbed (VINT) project. Currently the National Science Foundation (NSF) has joined the ride in development.

# Basic Architecture:

NS2 provides users with an executable command ns which takes on input argument, the name of a Tcl simulation scripting file. Users are feeding the name of a Tcl simulation script (which sets up a simulation) as an input argument of an NS2 executable command ns. In most cases, a simulation trace file is created, and is used to plot graph and/or to createanimation.

NS2 consists of two key languages:

* + C++
  + Object-oriented Tool Command Language(OTcl).



While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL. Mapped to a C++ object, variables in the OTcl domains are sometimes referred to as *handle*s. Conceptually, a handle is just a string in the OTcl domain, and does not contain any functionality. Instead, the functionality (e.g., receiving a packet) is defined in the mapped C++ object (e.g., of class Connector). In the OTcl domain, a handle acts as a frontend which interacts with users and other OTcl objects. It may

defines its own proceduresand variables to facilitate the interaction. Note that the member procedures and variables in the OTcl domain are called instance procedures (instprocs) and instance variables (instvars), respectively.After simulation, NS2 outputs either text-based or animation-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behavior of the network, users can extract a relevant subset of text-based data and transform it to a more conceivablepresentation.

# Network AniMation (NAM) Trace

NAM trace is records simulation detail in a text file, and uses the text file the play back the simulation using animation. NAM trace is activated by the command “$ns namtrace-all $file”, where ns is the Simulator handle and file is a handle associated with the file which stores the NAM trace information. After obtaining a NAM trace file, the animation can be initiated directly at the command prompt through the following command **>>nam filename.nam.** Many visualization features are available in NAM. These features are for example animating colored packet flows, dragging and dropping nodes (positioning), labeling nodes at a specified instant, shaping the nodes, coloring a specific link, and monitoring aqueue.

# NS2 Simulation Steps: Step 1: Simulation Design

In this step,the users should determine the simulation purposes, network configuration and assumptions, the performance measures, and the type of expected results.

# Step 2: Configuring and Running Simulation

This step implements the design in the first step. It consists of two phases:

* ***Network configuration phase*:** In this phase network components (e.g., node, TCP and UDP) are created and configured according to the simulation design. Also, the events such as data transfer are scheduled to start at a certaintime.
* ***Simulation Phase*:** This phase starts the simulation which was configured in the Network Configuration Phase. It maintains the simulation clock and executes events chronologically. This phase usually runs until the simulation clock reached a threshold value specified in the Network ConfigurationPhase.

In most cases, it is convenient to define a simulation scenario in a Tcl scripting file (e.g., <file>) and feed the file as an input argument of an NS2 invocation (e.g., executing “ns<file>”).

# Step 3: Post Simulation Processing

The main tasks in this steps include verifying the integrity of the program and evaluating the performance of the simulated network. While the first task is referred to as *debugging*, the second one is achieved by properly collecting and compiling simulation results.

# Performance Metrics of Routing Protocols:

In order to evaluate the performance of routing protocols, the following metrics were considered

**End-to-End Delay:**The average time interval between the generation of a packet in a source node and the successfully delivery of the packet at the destination node. It counts all possible delays that can occur inthesourceandallintermediatenodes,includingqueuingtime,packettransmissionandpropagation,and retransmissions at the MAC layer. The queuing time can be caused by network congestion or unavailability of valid routes.

**Packet Delivery Fraction:** The ratio of the number of data packets successfully delivered to all destination nodes and the number of data packets generated by all source nodes.

**Routing Load:** The ratio of the number of routing messages propagated by every node in the network and the number of data packets successfully delivered to all destination nodes. In other words, the routing load means the average number of routing messages generated to each data packet successfully delivered to the destination.

**Number of Packets dropped:** The number of data packets that are notsuccessfully sent to the destination during thetransmission.

**Jitter**:Jitter describes standard deviation of packet delay between all nodes.

**Throughput:**The throughput metric measures how well the network canconstantly provide data to the sink. Throughput is the number of packet arriving atthe sink per ms.

**Power consumption:**The total consumed energy divided by the number of delivered packet.

# Routing:

**Shortest Path Routing**

The following technique is widely used in many forms, because it is simple and easy to understand. The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc representing a communication line (link).To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.Different metrics like number of hops, geographical distance, the mean queuing and transmission delay of router can be used. The Dijkstra algorithm is used for computing shortestpath

# Flooding

That is another static algorithm, in with every incoming packet is sent out on every outgoing line except the one it arrived on. Flooding generates infinite number of duplicate packets unless some measures are taken to damp the process. One such measure is to have a hop counter in the header of each packet, which is decremented at each hop, with the packet being discarded when the counter reaches zero .Ideally, the hop counter is initialized to the length of the path from source to destination. If the sender does not no the path length, it can initialize the counter to the worst case, the full diameter of the subnet.

# Link State Routing

1. Discover its neighbors and learn their networkaddresses.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling to all it has justlearned.
4. Send the packet to all otherrouters.
5. Compute the shortest path(Using Dijkstra’s algorithm) to every otherrouter.

# Hierarchical Routing

As the network grows larger the amount of resources necessary to take care or routing table becomes enormous and makes routing impossible. Here appears the idea of hierarchical routing that suggests that routers should be divided into **regions**, with each router knowing all the details about how to route packets within its own region, but knowing nothing about the internal structure of other regions.Unfortunately the gains in routing table size & CPU time are not free, the penalty of increasing path length has to be paid.It has been discovered that the optimal number of nested levels for an N router subnet is *ln N*, requiring a total of *eln N* entries perrouter.

NS-2 has been widely used in sensor network simulations with extended libraries. NS2 is a simulation tool designed specifically for communication networks.The main functionalities of NS2 are to set up a network of connecting nodes and to pass packets from one node to another. A network object is one of the main NS2 components, which is responsible for packet forwarding. NS2 implements network objects by using the polymorphism concept in Object-Oriented Programming .A Node plays two important roles in NS2. As a router, it forwards packets to the connecting link based on a routing table. As a host, it delivers packets to the transport layer agent attached to the port specified in the packet header.NS2 configures the connection to its downstream NsObjects only. A Node does not need to have a connection to its upstream NsObject Instead, its upstream NsObject will create a connection to Node.

# Result:

Thus the study of implementation of routing using NS2 is done.

|  |  |
| --- | --- |
| **Ex. No:13.1** | **Stop and Wait Protocol** |
| **Date:** |

**AIM**

To write a java program to perform sliding window protocol

**ALGORITHM:**

1.Start the program.

2.Get the frame size from the user

3.To create the frame based on the user request. 4.To send frames

to server from the client side.

5.If your frames reach the server it will send ACK signal to client otherwise it will send NACK

signal to client.

6.Stop the program

**PROGRAM**

**SENDER PROGRAM**

import java.io.\*;

import java.net.\*;

public class Sender

{

Socket sender;

ObjectOutputStream out;

ObjectInputStream in;

String packet,ack,str, msg;

int n,i=0,sequence=0;

Sender(){}

public void run()

{

try{

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

System.out.println("Waiting for Connection....");

sender = new Socket("localhost",2004);

sequence=0;

out=new ObjectOutputStream(sender.getOutputStream());

out.flush();

in=new ObjectInputStream(sender.getInputStream());

str=(String)in.readObject();

System.out.println("reciver > "+str);

System.out.println("Enter the data to send....");

packet=br.readLine();

n=packet.length();

do{

try{

if(i<n){

msg=String.valueOf(sequence);

msg=msg.concat(packet.substring(i,i+1));

}

else if(i==n){

msg="end";out.writeObject(msg);break;

}

out.writeObject(msg);

sequence=(sequence==0)?1:0;

out.flush();

System.out.println("data sent>"+msg);

ack=(String)in.readObject();

System.out.println("waiting for ack.....\n\n");

if(ack.equals(String.valueOf(sequence))){

i++;

System.out.println("receiver > "+" packet recieved\n\n");

}

else{

System.out.println("Time out resending data....\n\n");

sequence=(sequence==0)?1:0;

}

}catch(Exception e){}

}while(i<n+1);

System.out.println("All data sent. exiting.");

}catch(Exception e){}

finally{

try{

in.close();

out.close();

sender.close();

}

catch(Exception e){}

}

}

public static void main(String args[])

{ Sender s=new Sender();

s.run();

}

}

**RECEIVER PROGRAM**

import java.io.\*;

import java.net.\*;

public class Reciever

{

ServerSocket reciever;

Socket connection=null;

ObjectOutputStream out;

ObjectInputStream in;

String packet,ack,data="";

int i=0,sequence=0;

Reciever(){}

public void run()

{

try{

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

reciever = new ServerSocket(2004,10);

System.out.println("waiting for connection...");

connection=reciever.accept();

sequence=0;

System.out.println("Connection established :");

out=new ObjectOutputStream(connection.getOutputStream());

out.flush();

in=new ObjectInputStream(connection.getInputStream());

out.writeObject("connected .");

do{

try{

packet=(String)in.readObject();

if(Integer.valueOf(packet.substring(0,1))==sequence){

data+=packet.substring(1);

sequence=(sequence==0)?1:0;

System.out.println("\n\nreceiver >"+packet);

}

else

{

System.out.println("\n\nreceiver >"+packet +" duplicate data");

}

if(i<3){

out.writeObject(String.valueOf(sequence));i++;

}

else{

out.writeObject(String.valueOf((sequence+1)%2));

i=0;

}

}

catch(Exception e){}

}while(!packet.equals("end"));

System.out.println("Data recived="+data);

out.writeObject("connection ended .");

}

catch(Exception e){}

finally{

try{

in.close();

out.close();

reciever.close();

}

catch(Exception e){}

}

}

public static void main(String args[]){

Reciever s=new Reciever();

while(true){

s.run();

}

}

}

|  |  |
| --- | --- |
| **Ex. No:13.2** | **Implementation of Sub netting** |
| **Date:** |

**Aim:**

Write a program to implement sub netting and find the subnet masks.

**Algorithm :**

1.Start the program.

2.Get the frame size from the user

3.To create the frame based on the user request.

4.To send frames to server from the client side.

5.If your frames reach the server it will send ACK signal to client otherwise it will

send NACK signal to client.

6.Stop the program

**Program**

import java.util.Scanner;

class Subnet

{

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

System.out.print(“Enter the ip address”);

String ip = sc.nextLine();

String split\_ip[]=ip.split(“\\.”);

//SPlit the string after every .

String split\_bip[] = new String[4];

//split binary ip

String bip = “”;

for(int i=0;i<4;i++)

{

split\_bip[i] = appendZeros(Integer.toBinaryString(Integer.parseInt(split\_ip[i])));

// “18” => 18 => 10010=>00010010

bip += split\_bip[i];

}

System.out.println(“IP in binary is “+bip);

System.out.print(“Enter the number of addresses: “);

int n = sc.nextInt();

int bits = (int)Math.ceil(Math.log(n)/Math.log(2));

System.out.println(“Number of bits required for address = “+bits); int mask = 32-bits;

System.out.println(“The subnet mask is = “ +mask);

address int fbip[] = new int[32];

for(int i=0; i<32;i++) fbip[i] = (int)bip.charAt(i)-48;

fbip[i] &= 0;

String fip[] = {“”,””,””,””};

for(int i=0;i<32;i++)

fip[i/8] = new String(fip[i/8]+fbip[i]);

System.out.print(“First address is = “);

for(int i=0;i<4;i++){

System.out.print(Integer.parseInt(fip[i],2));

if(i!=3) System.out.print(“.”);

}

System.out.println();

int lbip[] = new int[32];

for(int i=0; i<32;i++) lbip[i] = (int)bip.charAt(i)-48; //convert cahracter 0,1 to integer 0,1

for(int i=31;i>31-bits;i–) //Get last address by ORing last n bits with 1

lbip[i] |= 1;

String lip[] ={“”,””,””,””};

for(int i=0;i<32;i++)

lip[i/8] = new String(lip[i/8]+lbip[i]);

System.out.print(“Last address is =”);

for(int i=0;i<4;i++){

System.out.print(Integer.parseInt(lip[i],2));

if(i!=3) System.out.print(“.”);

}

System.out.println();

}

static String appendZeros(String s)

{ String temp = new String(“00000000″);

return temp.substring(s.length())+ s;

}

}

|  |  |
| --- | --- |
| **Ex. No:13.3** | **Implementation of TRACEROUTE** |
| **Date:** |

**Aim:**

To Write the java program for simulating Traceroute command

**Algorithm**

**Program**

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

import traceroute.TracerouteItem;

public abstract class Traceroute

{

private Runtime run;

public Traceroute()

{

run = Runtime.getRuntime();

}

public ArrayList<TracerouteItem> traceroute(String destination)

{

ArrayList<TracerouteItem> result = new ArrayList<TracerouteItem>();

Process pr = null;

String cmd = getTracerouteCommand(destination);

try

{

pr = run.exec(cmd);

}

catch(IOException e)

{

e.printStackTrace();

}

BufferedReader buf = new BufferedReader(new InputStreamReader(

pr.getInputStream()));

String line = "";

try

{

while((line = buf.readLine()) != null)

{

TracerouteItem item = parse(line);

result.add(item);

}

}

catch(IOException e)

{ return null;

}

return result;

}

public abstract TracerouteItem parse(String line);

public abstract String getTracerouteCommand(String destination);

}

|  |  |
| --- | --- |
| **Ex. No:13.4** | **IMPLEMENTATION of DNS** |
| **Date:** |

**Aim**

To write a java program for DNS application

**Algorithm**

1.Start the program.

2.Get the frame size from the user

3.To create the frame based on the user request.

4.To send frames to server from the client side.

5.If your frames reach the server it will send ACK signal to client otherwise it will

send NACK signal to client.

6.Stop the program

**Program**

**// UDP DNS Server**

**Udpdnsserver**

**.**java import java.io.\*;

import java.net.\*;

public class udpdnsserver

{

private static int indexOf(String[] array, String str)

{

str = str.trim();

for (int i=0; i < array.length; i++)

{

if (array[i].equals(str)) return i;

}

return -1;

}

public static void main(String arg[])throws IOException

{

String[] hosts = {"yahoo.com", "gmail.com","cricinfo.com", "facebook.com"};

String[] ip = {"68.180.206.184", "209.85.148.19","80.168.92.140", "69.63.189.16"};

System.out.println("Press Ctrl + C to Quit");

while (true)

{

DatagramSocket serversocket=new DatagramSocket(1362);

byte[] senddata = new byte[1021];

byte[] receivedata = new byte[1021];

DatagramPacket recvpack = new DatagramPacket(receivedata, receivedata.length);

serversocket.receive(recvpack);

String sen = new String(recvpack.getData());

InetAddress ipaddress = recvpack.getAddress();

int port = recvpack.getPort();

String capsent;

System.out.println("Request for host " + sen);

if(indexOf (hosts, sen) != -1)

capsent = ip[indexOf (hosts, sen)];

else capsent = "Host Not Found";

senddata = capsent.getBytes();

DatagramPacket pack = new DatagramPacket (senddata, senddata.length,ipaddress,port);

serversocket.send(pack);

serversocket.close();

}

}

}

//**UDP DNS Client** –

Udpdnsclient

.java import java.io.\*;

import java.net.\*;

public class udpdnsclient

{

public static void main(String args[])throws IOException

{

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

DatagramSocket clientsocket = new DatagramSocket();

InetAddress ipaddress;

if (args.length == 0)

ipaddress =

InetAddress.getLocalHost(); else

ipaddress = InetAddress.getByName(args[0]);

byte[] senddata = new byte[1024];

byte[] receivedata = new byte[1024];

int portaddr = 1362;

System.out.print("Enter the hostname : ");

String sentence = br.readLine();

Senddata = sentence.getBytes();

DatagramPacket pack = new DatagramPacket(senddata,senddata.length, ipaddress,portaddr);

clientsocket.send(pack);

DatagramPacket recvpack =new DatagramPacket(receivedata,receivedata.length);

clientsocket.receive(recvpack);

String modified = new String(recvpack.getData());

System.out.println("IP Address: " + modified);

clientsocket.close();

}

}

|  |  |
| --- | --- |
| **Ex. No:13.5** | **SNMP** |
| **Date:** |

**Aim :**To write a java program for SNMP application program

**Algorithm**

1.Start the program.

2.Get the frame size from the user

3.To create the frame based on the user request.

4.To send frames to server from the client side.

5.If your frames reach the server it will send ACK signal to client otherwise it will

send NACK signal to client.

6.Stop the program

**Program**

import java.io.IOException;

import org.snmp4j.CommunityTarget;

import org.snmp4j.PDU;

import org.snmp4j.Snmp;

import org.snmp4j.Target;

import org.snmp4j.TransportMapping;

import org.snmp4j.event.ResponseEvent;

import org.snmp4j.mp.SnmpConstants;

import org.snmp4j.smi.Address;

import org.snmp4j.smi.GenericAddress;

import org.snmp4j.smi.OID;

import org.snmp4j.smi.OctetString;

import org.snmp4j.smi.VariableBinding;

import org.snmp4j.transport.DefaultUdpTransportMapping;

public class SNMPManager

{

Snmp snmp = null;

String address = null;

public SNMPManager(String add)

{

address = add;

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

SNMPManager client = new SNMPManager("udp:127.0.0.1/161");

client.start();

String sysDescr = client.getAsString(new OID(".1.3.6.1.2.1.1.1.0"));

System.out.println(sysDescr);

}

private void start() throws IOException

{

TransportMapping transport = new DefaultUdpTransportMapping();

snmp = new

Snmp(transport);

transport.listen();

}

public String getAsString(OID oid) throws IOException

{

ResponseEvent event = get(new OID[] { oid });

return event.getResponse().get(0).getVariable().toString();

}

public ResponseEvent get(OID oids[]) throws IOException

{

PDU pdu = new PDU();

for (OID oid : oids)

{

pdu.add(new VariableBinding(oid));

}

pdu.setType(PDU.GET);

ResponseEvent event = snmp.send(pdu, getTarget(), null);

if(event != null)

{

return event;

}

throw new RuntimeException("GET timed out");

}

private Target getTarget()

{

Address targetAddress = GenericAddress.parse(address);

CommunityTarget target = new CommunityTarget();

target.setCommunity(new OctetString("public"));

target.setAddress(targetAddress);

target.setRetries(2);

target.setTimeout(1500);

target.setVersion(SnmpConstants.version2c);

return target;

}}

|  |  |
| --- | --- |
| **Ex. No:14** | **VIVA QUESTIONS** |
| **Date:** |

**1. Define Network?**

A network is a set of devices connected by physical media links. A network is recursively is a connection of

two or more nodes by a physical link or two or more networks connected by one or more nodes.

**2. What is a Link?**

At the lowest level, a network can consist of two or more computers directly connected by some physical medium such as coaxial cable or optical fiber. Such a physical medium is called as Link.

**3. What is a node?**

A network can consist of two or more computers directly connected by some physical medium such as coaxial cable or optical fiber. Such a physical medium is called as Links and the computer it connects is called as Nodes.

**4. What is a gateway or Router?**

A node that is connected to two or more networks is commonly called as router or Gateway. It generally forwards message from one network to another.

**5. What is point-point link?**

If the physical links are limited to a pair of nodes it is said to be point-point link.

**6. What is Multiple Access?**

If the physical links are shared by more than two nodes, it is said to be Multiple Access.

**7. What are the advantages of Distributed Processing?**

a. Security/Encapsulation  
b. Distributed database  
c. Faster Problem solving  
d. Security through redundancy  
e. Collaborative Processing

**8. What are the criteria necessary for an effective and efficient network?**

a. Performance -    It can be measured in many ways, including transmit time and response time.

b. Reliability -    It is measured by frequency of failure, the time it takes a link to recover from a failure,

and the network's robustness.  
c. Security -   Security issues includes protecting data from unauthorized access and virues.

**9. Name the factors that affect the performance of the network?**

a. Number of Users  
b. Type of transmission medium  
c. Hardware  
d. Software

**10. Name the factors that affect the reliability of the network?**

a. Frequency of failure  
b. Recovery time of a network after a failure

**11. Name the factors that affect the security of the network?**

a. Unauthorized Accessb. Viruses

**12. What is Protocol?**

A protocol is a set of rules that govern all aspects of information communication.

**13. What are the key elements of protocols?**

The key elements of protocols are  
a. Syntax  
   It refers to the structure or format of the data, that is the order in which they are presented.  
b. Semantics  
   It refers to the meaning of each section of bits.  
c. Timing  
   Timing refers to two characteristics: When data should be sent and how fast they can be sent.

**14. What are the key design issues of a computer Network?**

a. Connectivity  
b. Cost-effective Resource Sharing  
c. Support for common Services  
d. Performance

**15. Define Bandwidth and Latency?**

Network performance is measured in Bandwidth (throughput) and Latency (Delay). Bandwidth of a network is given by the number of bits that can be transmitted over the network in a certain period of time. Latency corresponds to how long it t5akes a message to travel from one end off a network to the other. It is strictly measured in terms of time.

**16. Define Routing?**

The process of determining systematically hoe to forward messages toward the destination nodes based on its address is called routing.

**17. What is a peer-peer process?**

The processes on each machine that communicate at a given layer are called peer-peer process.

**18. When a switch is said to be congested?**

It is possible that a switch receives packets faster than the shared link can accommodate and stores in its memory, for an extended period of time, then the switch will eventually run out of buffer space, and some packets will have to be dropped and in this state is said to congested state.

**19. What is semantic gap?**

Defining a useful channel involves both understanding the applications requirements and recognizing the limitations of the underlying technology. The gap between what applications expects and what the underlying technology can provide is called semantic gap.

**20. What is Round Trip Time?**

The duration of time it takes to send a message from one end of a network to the other and back, is called RTT.

**21. Define the terms Unicasting, Multiccasting and Broadcasting?**

If the message is sent from a source to a single destination node, it is called Unicasting.  
If the message is sent to some subset of other nodes, it is called Multicasting.  
If the message is sent to all the m nodes in the network it is called Broadcasting.

**22. What is Multiplexing?**

Multiplexing is the set of techniques that allows the simultaneous transmission of multiple signals across a single data link.

**23. Name the categories of Multiplexing?**

a. Frequency Division Multiplexing (FDM)  
b. Time Division Multiplexing (TDM)  
   i. Synchronous TDM  
   ii. ASynchronous TDM Or Statistical TDM.  
c. Wave Division Multiplexing (WDM)

**24. What is FDM?**

FDM is an analog technique that can be applied when the bandwidth of a link is greater than the combined bandwidths of the signals to be transmitted.

**25. What is WDM?**

WDM is conceptually the same as FDM, except that the multiplexing and demultiplexing involve light signals transmitted through fiber optics channel.

**26. What is TDM?**

TDM is a digital process that can be applied when the data rate capacity of the transmission medium is greater than the data rate required by the sending and receiving devices.

**27. What is Synchronous TDM?**

In STDM, the multiplexer allocates exactly the same time slot to each device at all times, whether or not a device has anything to transmit.

**28. List the layers of OSI**

a. Physical Layer  
b. Data Link Layer  
c. Network Layer  
d. Transport Layer  
e. Session Layer  
f. Presentation Layer  
g. Application Layer

**29. Which layers are network support layers?**

a. Physical Layer  
b. Data link Layer and   
c. Network Layers

**30. Which layers are user support layers?**

a. Session Layer  
b. Presentation Layer and   
c. Application Layer

**31. Which layer links the network support layers and user support layers?**

The Transport layer links the network support layers and user support layers.

**32. What are the concerns of the Physical Layer?**

Physical layer coordinates the functions required to transmit a bit stream over a physical medium.  
a. Physical characteristics of interfaces and media  
b. Representation of bits  
c. Data rate  
d. Synchronization of bits  
e. Line configuration  
f. Physical topology  
g. Transmission mode

**33. What are the responsibilities of Data Link Layer?**

The Data Link Layer transforms the physical layer, a raw transmission facility, to a reliable link and is responsible for node-node delivery.  
a. Framing  
b. Physical Addressing  
c. Flow Control  
d. Error Control  
e. Access Control

**34. What are the responsibilities of Network Layer?**

The Network Layer is responsible for the source-to-destination delivery of packet possibly across multiple networks (links).  
a. Logical Addressing  
b. Routing

**35. What are the responsibilities of Transport Layer?**

The Transport Layer is responsible for source-to-destination delivery of the entire message.  
a. Service-point Addressing  
b. Segmentation and reassembly  
c. Connection Control  
d. Flow Control  
e. Error Control

**36. What are the responsibilities of Session Layer?**

The Session layer is the network dialog Controller. It establishes, maintains and synchronizes the interaction between the communicating systems.  
a. Dialog control  
b. Synchronization

**37. What are the responsibilities of Presentation Layer?**

The Presentation layer is concerned with the syntax and semantics of the information exchanged between two systems.  
a. Translation  
b. Encryption  
c. Compression

**38. What are the responsibilities of Application Layer?**

The Application Layer enables the user, whether human or software, to access the network. It provides user interfaces and support for services such as e-mail, shared database management and other types of distributed information services.  
a. Network virtual Terminal  
b. File transfer, access and Management (FTAM)  
c. Mail services  
d. Directory Services

**39. What are the two classes of hardware building blocks?**

Nodes and Links.

**40. What are the different link types used to build a computer network?**

a. Cables  
b. Leased Lines  
c. Last-Mile Links  
d. Wireless Links

**41. What are the categories of Transmission media?**

a. Guided Media  
  i. Twisted - Pair cable  
    1. Shielded TP  
    2. Unshielded TP  
  ii. Coaxial Cable  
  iii. Fiber-optic cable  
b. Unguided Media  
  i. Terrestrial microwave  
  ii. Satellite Communication

**42. What are the types of errors?**

a. Single-Bit error  
  In a single-bit error, only one bit in the data unit has changed  
b. Burst Error  
  A Burst error means that two or more bits in the data have changed.

**43. What is Error Detection? What are its methods?**

Data can be corrupted during transmission. For reliable communication errors must be deducted and Corrected. Error Detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination. The common Error Detection methods are   
  a. Vertical Redundancy Check (VRC)  
  b. Longitudinal Redundancy Check (VRC)  
  c. Cyclic Redundancy Check (VRC)  
  d. Checksum

**44. What is Redundancy?**

The concept of including extra information in the transmission solely for the purpose of comparison. This technique is called redundancy.

**45. What is VRC?**

It is the most common and least expensive mechanism for Error Detection. In VRC, a parity bit is added to every data unit so that the total number of 1s becomes even for even parity. It can detect all single-bit errors. It can detect burst errors only if the total number of errors in each data unit is odd.

**46. What is LRC?**

In LRC, a block of bits is divided into rows and a redundant row of bits is added to the whole block. It can detect burst errors. If two bits in one data unit are damaged and bits in exactly the same positions in another data unit are also damaged, the LRC checker will not detect an error. In LRC a redundant data unit follows n data units.

**47. What is CRC?**

CRC, is the most powerful of the redundancy checking techniques, is based on binary division.

**48. What is Checksum?**

Checksum is used by the higher layer protocols (TCP/IP) for error detection

**49. List the steps involved in creating the checksum.**

a. Divide the data into sections  
b. Add the sections together using 1's complement arithmetic  
c. Take the complement of the final sum, this is the checksum.

**50. What are the Data link protocols?**

Data link protocols are sets of specifications used to implement the data link layer. The categories of Data Link protocols are

1. Asynchronous Protocols  
   2. Synchronous Protocols  
     a. Character Oriented Protocols  
     b. Bit Oriented protocols

**51. Compare Error Detection and Error Correction:**

The correction of errors is more difficult than the detection. In error detection, checks only any error has occurred. In error correction, the exact number of bits that are corrupted and location in the message are known. The number of the errors and the size of the message are important factors.

**52. What is Forward Error Correction?**

Forward error correction is the process in which the receiver tries to guess the message by using redundant bits.

**53. Define Retransmission?**

Retransmission is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-freed.

**54. What are Data Words?**

In block coding, we divide our message into blocks, each of k bits, called datawords. The block coding process is one-to-one. The same dataword is always encoded as the same codeword.

**55. What are Code Words?**

"r" redundant bits are added to each block to make the length n = k + r. The resulting n-bit blocks are called codewords. 2n - 2k codewords that are not used. These codewords are invalid or illegal.

**56. What is a Linear Block Code?**

A linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

**57. What are Cyclic Codes?**

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

**58. Define Encoder?**

A device or program that uses predefined algorithms to encode, or compress audio or video data for storage or transmission use. A circuit that is used to convert between digital video and analog video.

**59. Define Decoder?**

A device or program that translates encoded data into its original format (e.g. it decodes the data). The term is often used in reference to MPEG-2 video and sound data, which must be decoded before it is output.

**60. What is Framing?**

Framing in the data link layer separates a message from one source to a destination, or from other messages to other destinations, by adding a sender address and a destination address. The destination address defines where the packet has to go and the sender address helps the recipient acknowledge the receipt.

**61. What is Fixed Size Framing?**

In fixed-size framing, there is no need for defining the boundaries of the frames. The size itself can be used as a delimiter.

**62. Define Character Stuffing?**

In byte stuffing (or character stuffing), a special byte is added to the data section of the frame when there is a character with the same pattern as the flag. The data section is stuffed with an extra byte. This byte is usually called the escape character (ESC), which has a predefined bit pattern. Whenever the receiver encounters the ESC character, it removes it from the data section and treats the next character as data, not a delimiting flag.

**63. What is Bit Stuffing?**

Bit stuffing is the process of adding one extra 0 whenever five consecutive Is follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

**64. What is Flow Control?**

Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.

**65. What is Error Control ?**

Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and retransmission.

**66. What Automatic Repeat Request (ARQ)?**

Error control is both error detection and error correction. It allows the receiver to inform the sender of any frames lost or damaged in transmission and coordinates the retransmission of those frames by the sender. In the data link layer, the term error control refers primarily to methods of error detection and retransmission. Error control in the data link layer is often implemented simply: Any time an error is detected in an exchange, specified frames are retransmitted. This process is called automatic repeat request (ARQ).

**67. What is Stop-and-Wait Protocol?**

In Stop and wait protocol, sender sends one frame, waits until it receives confirmation from the receiver (okay to go ahead), and then sends the next frame.

**68. What is Stop-and-Wait Automatic Repeat Request?**

Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

**69. What is usage of Sequence Number in Relaible Transmission?**

The protocol specifies that frames need to be numbered. This is done by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame. Since we want to minimize the frame size, the smallest range that provides unambiguous communication. The sequence numbers can wrap around.

**70. What is Pipelining ?**

In networking and in other areas, a task is often begun before the previous task has ended. This is known as pipelining.

**71. What is Sliding Window?**

The sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, he sender and receiver need to deal with only part of the possible sequence numbers.

**72. What is Piggy Backing?**

A technique called piggybacking is used to improve the efficiency of the bidirectional protocols. When a frame is carrying data from A to B, it can also carry control information about arrived (or lost) frames from B; when a frame is carrying data from B to A, it can also carry control information about the arrived (or lost) frames from A.

**73. What are the two types of transmission technology available?**

(i) Broadcast and (ii) point-to-point

**74. What is subnet?**

A generic term for section of a large networks usually separated by a bridge or router.

**75. Difference between the communication and transmission.**

Transmission is a physical movement of information and concern issues like bit polarity, synchronization, clock etc.

Communication means the meaning full exchange of information between two communication media.

**76. What are the possible ways of data exchange?**

(i) Simplex (ii) Half-duplex (iii) Full-duplex.

**77. What is SAP?**

Series of interface points that allow other computers to communicate with the other layers of network

protocol stack.

**78. What do you meant by "triple X" in Networks?**

The function of PAD (Packet Assembler Disassembler) is described in a document known as X.3.

The standard protocol has been defined between the terminal and the PAD, called X.28; another

standard protocol exists between hte PAD and the network, called X.29. Together, these three recommendations are often called "tripleX".

**79. What is frame relay, in which layer it comes?**

Frame relay is a packet switching technology. It will operate in the data link layer.

**80. What is terminal emulation, in which layer it comes?**

Telnet is also called as terminal emulation. It belongs to application layer.

**81. What is Beaconing?**

The process that allows a network to self-repair networks problems. The stations on the network notify the other stations on the ring when they are not receiving the transmissions. Beaconing is used in Token ring and FDDI networks.

**82. What is redirector?**

Redirector is software that intercepts file or prints I/O requests and translates them into network requests. This comes under presentation layer.

**83. What is NETBIOS and NETBEUI?**

NETBIOS is a programming interface that allows I/O requests to be sent to and received from a remote computer and it hides the networking hardware from applications.

NETBEUI is NetBIOS extended user interface. A transport protocol designed by microsoft and IBM for the use on small subnets.

**84. What is RAID?**

A method for providing fault tolerance by using multiple hard disk drives.

**85. What is passive topology?**

When the computers on the network simply listen and receive the signal, they are referred to as passive because they don't amplify the signal in any way. Example for passive topology -linear bus.

**86. What is Brouter?**

Hybrid devices that combine the features of both bridges and routers.

**87. What is cladding?**

A layer of a glass surrounding the center fiber of glass inside a fiber-optic cable.

**88. What is point-to-point protocol?**

A communications protocol used to connect computers to remote networking services including Internet service providers.

**89. How Gateway is different from Routers?**

A gateway operates at the upper levels of the OSI model and translates information between two completely different network architectures or data formats.

**90. What is attenuation?**

The degeneration of a signal over distance on a network cable is called attenuation.

**91. What is MAC address?**

The address for a device as it is identified at the Media Access Control (MAC) layer in the network architecture. MAC address is usually stored in ROM on the network adapter card and is unique.

**92. Difference between bit rate and baud rate.**

Bit rate is the number of bits transmitted during one second whereas baud rate refers to the number of signal units per second that are required to represent those bits.  
  baud rate = (bit rate / N)

where N is no-of-bits represented by each signal shift.

**93. What is Bandwidth?**

Every line has an upper limit and a lower limit on the frequency of signals it can carry. This limited range is called the bandwidth.

**94. What are the types of Transmission media?**

Signals are usually transmitted over some transmission media that are broadly classified in to two categories.

a.) **Guided Media**: These are those that provide a conduit from one device to another that include twisted-pair, coaxial cable and fiber-optic cable. A signal traveling along any of these media is directed and is contained by the physical limits of the medium. Twisted-pair and coaxial cable use metallic that accept and transport signals in the form of electrical current. Optical fiber is a glass or plastic cable that accepts and transports signals in the form of light.

b.) **Unguided Media**: This is the wireless media that transport electromagnetic waves without using a physical conductor. Signals are broadcast either through air. This is done through radio communication, satellite communication and cellular telephony.

**95. What is Project 802?**

It is a project started by IEEE to set standards to enable intercommunication between equipment from a variety of manufacturers. It is a way for specifying functions of the physical layer, the data link layer and to some extent the network layer to allow for interconnectivity of major LAN protocols.

It consists of the following:

1.    802.1 is an internetworking standard for compatibility of different LANs and MANs across protocols.

2.    802.2 Logical link control (LLC) is the upper sublayer of the data link layer which is non-architecture-specific, that is remains the same for all IEEE-defined LANs.

3.    Media access control (MAC) is the lower sublayer of the data link layer that contains some distinct modules each carrying proprietary information specific to the LAN product being used. The modules are Ethernet LAN (802.3), Token ring LAN (802.4), Token bus LAN (802.5).

4.    802.6 is distributed queue dual bus (DQDB) designed to be used in MANs.

**96. What is Protocol Data Unit?**

The data unit in the LLC level is called the protocol data unit (PDU). The PDU contains of four fields a destination service access point (DSAP), a source service access point (SSAP), a control field and an information field. DSAP, SSAP are addresses used by the LLC to identify the protocol stacks on the receiving and sending machines that are generating and using the data. The control field specifies whether the PDU frame is a information frame (I - frame) or a supervisory frame (S - frame) or a unnumbered frame (U - frame).

**97. What are the different type of networking / internetworking devices?**

1.    **Repeater**: Also called a regenerator, it is an electronic device that operates only at physical layer. It receives the signal in the network before it becomes weak, regenerates the original bit pattern and puts the refreshed copy back in to the link.

2.    **Bridges**: These operate both in the physical and data link layers of LANs of same type. They divide a larger network in to smaller segments. They contain logic that allow them to keep the traffic for each segment separate and thus are repeaters that relay a frame only the side of the segment containing the intended recipent and control congestion.

3.    **Routers**: They relay packets among multiple interconnected networks (i.e. LANs of different type). They operate in the physical, data link and network layers. They contain software that enable them to determine which of the several possible paths is the best for a particular transmission.

4.    **Gateways**: They relay packets among networks that have different protocols (e.g. between a LAN and a WAN). They accept a packet formatted for one protocol and convert it to a packet formatted for another protocol before forwarding it. They operate in all seven layers of the OSI model.

**98. What is ICMP?**

ICMP is Internet Control Message Protocol, a network layer protocol of the TCP/IP suite used by hosts and gateways to send notification of datagram problems back to the sender. It uses the echo test / reply to test whether a destination is reachable and responding. It also handles both control and error messages.

**99. What are the data units at different layers of the TCP / IP protocol suite?**

The data unit created at the application layer is called a message, at the transport layer the data unit created is called either a segment or an user datagram, at the network layer the data unit created is called the datagram, at the data link layer the datagram is encapsulated in to a frame and finally transmitted as signals along the transmission media.

**100. What is difference between ARP and RARP?**

The address resolution protocol (ARP) is used to associate the 32 bit IP address with the 48 bit physical address, used by a host or a router to find the physical address of another host on its network by sending a ARP query packet that includes the IP address of the receiver.

The reverse address resolution protocol (RARP) allows a host to discover its Internet address when it knows only its physical address.

**101. What is the minimum and maximum length of the header in the TCP segment and IP datagram?**

The header should have a minimum length of 20 bytes and can have a maximum length of 60 bytes.

**102. What is the range of addresses in the classes of internet addresses?**

Class A   -       0.0.0.0   -   127.255.255.255  
Class B   -   128.0.0.0   -   191.255.255.255  
Class C   -   192.0.0.0   -   223.255.255.255  
Class D   -   224.0.0.0   -   239.255.255.255  
Class E   -   240.0.0.0   -   247.255.255.255

**103. What is the difference between TFTP and FTP application layer protocols?**

The Trivial File Transfer Protocol (TFTP) allows a local host to obtain files from a remote host but does not provide reliability or security. It uses the fundamental packet delivery services offered by UDP.

The File Transfer Protocol (FTP) is the standard mechanism provided by TCP / IP for copying a file from one host to another. It uses the services offer by TCP and so is reliable and secure. It establishes two connections (virtual circuits) between the hosts, one for data transfer and another for control information.

**104. What are major types of networks and explain?**

1.    **Server-based network**: provide centralized control of network resources and rely on server computers to provide security and network administration

2.    **Peer-to-peer network**: computers can act as both servers sharing resources and as clients using the resources.

**105. What are the important topologies for networks?**

1.    **BUS topology**: In this each computer is directly connected to primary network cable in a single line.  
Advantages: Inexpensive, easy to install, simple to understand, easy to extend.

2.    **STAR topology**: In this all computers are connected using a central hub.  
Advantages: Can be inexpensive, easy to install and reconfigure and easy to trouble shoot physical problems.

3.    **RING topology**: In this all computers are connected in loop. Advantages: All computers have equal access to network media, installation can be simple, and signal does not degrade as much as in other topologies because each computer regenerates it.

**106. What is mesh network?**

A network in which there are multiple network links between computers to provide multiple paths for data to travel.

**107. What is difference between baseband and broadband transmission?**

In a baseband transmission, the entire bandwidth of the cable is consumed by a single signal. In broadband transmission, signals are sent on multiple frequencies, allowing multiple signals to be sent simultaneously.

**108. Explain 5-4-3 rule?**

In a Ethernet network, between any two points on the network ,there can be no more than five network segments or four repeaters, and of those five segments only three of segments can be populated.

**109. What MAU?**

In token Ring , hub is called Multi station Access Unit(MAU).

**110. What is the difference between routable and non- routable protocols?**

Routable protocols can work with a router and can be used to build large networks. Non-Routable protocols are designed to work on small, local networks and cannot be used with a router.

**111. Why should you care about the OSI Reference Model?**

It provides a framework for discussing network operations and design.

**112. What is logical link control?**

One of two sublayers of the data link layer of OSI reference model, as defined by the IEEE 802 standard. This sublayer is responsible for maintaining the link between computers when they are sending data across the physical network connection.

**113. What is virtual channel?**

Virtual channel is normally a connection from one source to one destination, although multicast connections are also permitted. The other name for virtual channel is virtual circuit.

**114. What is virtual path?**

Along any transmission path from a given source to a given destination, a group of virtual circuits can be grouped together into what is called path.

**115. What is packet filter?**

Packet filter is a standard router equipped with some extra functionality. The extra functionality allows every incoming or outgoing packet to be inspected. Packets meeting some criterion are forwarded normally. Those that fail the test are dropped.

**116. What is traffic shaping?**

One of the main causes of congestion is that traffic is often busy. If hosts could be made to transmit at a uniform rate, congestion would be less common. Another open loop method to help manage congestion is forcing the packet to be transmitted at a more predictable rate. This is called traffic shaping.

**117. What is multicast routing?**

Sending a message to a group is called multicasting, and its routing algorithm is called multicast routing.

**118. What is region?**

When hierarchical routing is used, the routers are divided into what we will call regions, with each router knowing all the details about how to route packets to destinations within its own region, but knowing nothing about the internal structure of other regions.

**119. What is silly window syndrome?**

It is a problem that can ruin TCP performance. This problem occurs when data are passed to the sending TCP entity in large blocks, but an interactive application on the receiving side reads 1 byte at a time.

**120. What are Digrams and Trigrams?**

The most common two letter combinations are called as digrams. e.g. th, in, er, re and an. The most common three letter combinations are called as trigrams. e.g. the, ing, and, and ion.

**121. Expand IDEA.**

IDEA stands for International Data Encryption Algorithm.

**122. What is wide-mouth frog?**

Wide-mouth frog is the simplest known key distribution center (KDC) authentication protocol.

**123. What is Mail Gateway?**

It is a system that performs a protocol translation between different electronic mail delivery protocols.

**124. What is IGP (Interior Gateway Protocol)?**

It is any routing protocol used within an autonomous system.

**125. What is EGP (Exterior Gateway Protocol)?**

It is the protocol the routers in neighboring autonomous systems use to identify the set of networks that can be reached within or via each autonomous system.

**126. What is autonomous system?**

It is a collection of routers under the control of a single administrative authority and that uses a common Interior Gateway Protocol.

**127. What is BGP (Border Gateway Protocol)?**

It is a protocol used to advertise the set of networks that can be reached with in an autonomous system. BGP enables this information to be shared with the autonomous system. This is newer than EGP (Exterior Gateway Protocol).

**128. What is Gateway-to-Gateway protocol?**

It is a protocol formerly used to exchange routing information between Internet core routers.

**129. What is NVT (Network Virtual Terminal)?**

It is a set of rules defining a very simple virtual terminal interaction. The NVT is used in the start of a Telnet session.

**130. What is a Multi-homed Host?**

It is a host that has a multiple network interfaces and that requires multiple IP addresses is called as a Multi-homed Host.

**131. What is Kerberos?**

It is an authentication service developed at the Massachusetts Institute of Technology. Kerberos uses encryption to prevent intruders from discovering passwords and gaining unauthorized access to files.

**132. What is OSPF?**

It is an Internet routing protocol that scales well, can route traffic along multiple paths, and uses knowledge of an Internet's topology to make accurate routing decisions.

**133. What is Proxy ARP?**

It is using a router to answer ARP requests. This will be done when the originating host believes that a destination is local, when in fact is lies beyond router.

**134. What is SLIP (Serial Line Interface Protocol)?**

It is a very simple protocol used for transmission of IP datagrams across a serial line.

**135. What is RIP (Routing Information Protocol)?**

It is a simple protocol used to exchange information between the routers.

**136. What is source route?**

It is a sequence of IP addresses identifying the route a datagram must follow. A source route may optionally be included in an IP datagram header.

**137. What is piggybacking ?**

Piggybacking on Internet access is the practice of establishing a wireless Internet connection by using another subscriber's wireless Internet access service without the subscriber's explicit permission or knowledge. It is a legally and ethically controversial practice, with laws that vary by jurisdiction around the world. While completely outlawed or regulated in some places, it is permitted in others.

**138. Name the two sub layers of Data link layer. Specify their protocols.**

1.  Logical link control(LLC)

Protocols:SDLC, NetBIOS, NetWare

2.  Media access Control (MAC)

Protocols :CSMA/CA, Slotted-ALOHA, CDMA, OFDMA

**139. How we arrived 7 layers of OSI reference model? Why not less than 7 or more than 7 ?**

The ISO looked to create a simple model for networking. They took the approach of defining layers that rest in a stack formation, one layer upon the other. Each layer would have a specific function, and deal with a specific task. Much time was spent in creating their model called "The ISO OSI Seven Layer Model for Networking". In this model, they have 7 layers, and each layer has a special and specific function.

**140.Why network standardization is done ?**

Computer networking is a great way of connecting the computers and sharing data with each other.  There are many vendors that produce different hardware devices and software applications and without coordination among them there can be chaos, unmanaged communication and disturbance can be faced by the users. There should be some rules and regulations that all the vendors should adopt and produce the devices basedon those communication standards. 

**141.Difference between unacknowledged connection less services and acknowledged connection less services ?**

 Unacknowledged connectionless service consists of having the source machine send independent frames

to the destination machine without having the destination machine acknowledged. Most LAN's use this service.

 Acknowledged connectionless service in this service there are no logical connections used but each frame

sent individually acknowledged. In this way the sender knows whether a frame has arrived correctly.

It is useful on wireless systems

**142.What is frame ?**

In computer networking and telecommunication, a frame is a digital data transmission unit or data packet

that includes frame synchronization, i.e. a sequence of bits or symbols making it possible for the

receiver to detect the beginning and end of the packet in the stream of symbols or bits.

**143.What is bit stuffing?**

Ans: Bit stuffing is the insertion of one or more bits into a transmission unit as a way to provide signaling information to a receiver. The receiver knows how to detect and remove or disregard the stuffed bits.

**144.What is encryption and decryption ?**

Ans: Encryption is an algorithm which converts the message into a form that is unreadable known as scrambled message and decryption is the process which converts the encrypted message into readable form known as unscrambled message. Actually this is a method to transfer message from one side to other in a secure manner.

**145.What are the various types of key used in cryptography ?**

Ans: here are two main types of cryptography:

* Secret key cryptography
* Public key cryptography

**Secret key cryptography** is also known as symmetric key cryptography. With this type of cryptography,

both the sender and the receiver know the same secret code, called the key. Messages are encrypted

by the sender using the key and decrypted by the receiver using the same key.

**Public key cryptography** also called asymmetric encryption, uses a pair of keys for encryption

and decryption. With public key cryptography, keys work in pairs of matched public and private keys.

 --------------------------------------------------------------------------------------------------------------------------------