Aim: To find Ip address and host name of the machine.
Algorithm:- Step 1: Import the necessary Java package. Step 2: Create a class. Get TD address. Step 3: Create an object addy for the class InetAddress and call getLocalHost. Step 4: Print the HostAddress of the machaine by calling getHostAddress() method. Step 5: Call getHostName() method to obtained the name of the machine and store in the string variable. Step6- Print the Local Hast Name. Step7: Terminate the program.
Program:-

```
import java.net.*;
public class GetIpAddress{
public static void main(String args[])throws Exception
{
InetAddress addr=InetAddress.getLocalHost();
System.out.println("Local Host Address: "+addr.getHostAddress());
String hostname=addr.getHostName();
System.out.println("Local Host Name: "+hostname);
}
}
```

Aim: Program to print date and time in client by server.
Alogorithm: SERVER STEP1: Create instances for socket and ServerSocket class.STEP2: Use the port 8020 for TCP.STEP3: Make the PrintStream object connect to the OuputStream using Socket.STEP4: Create an instance of the Date class and write it into the Socket. STEP5: Get the IP address of the client using the socket and getInetAddress(). CLIENT STEP1: Create instances for socket class with the port number 8020.STEP2: Create an object of DataInputStream and make it to get data from server through the socket. STEP3: Read the Date object.STEP4: Print the obtained date.
Program:
server:

```
import java.net.*;
import java.io.*;
import java.util.*;
public class server2 extends Thread
{
public static void main(String[] args) throws Exception
{
ServerSocket sSocket = new ServerSocket(1000);
Socket cSocket=sSocket.accept();
BufferedReader br=new BufferedReader(new
InputStreamReader(cSocket.getInputStream()));
PrintWriter out=new PrintWriter(cSocket.getOutputStream(),true);
Date d = new Date();
try
{
while(true)
{
d= new Date();
out.println("Time at server;" +d.toString());
System.out.println(br.readLine());
sleep(1000);
}
}
catch(IOException e)
{
System.out.println("----Client has Closed-----");
}
}
}
```

Client:

```
import java.net.*;
import java.io.*;
public class client2
{
public static void main (String[] arg) throws Exception
{
try
{
Socket s=new Socket(InetAddress.getLocalHost(),1000);
BufferedReader br = new BufferedReader(new
InputStreamReader(s.getInputStream()));
String input;
PrintWriter out=new PrintWriter(s.getOutputStream(),true);
while((input=br.readLine())!=null)
{
System.out.println(input);
out.println("Date and Time Received-----------client Acknowledge------");
}
}
catch(Exception e)
{
}
}
}
```

Aim: Program to print client address at server side.
Algorithm: SERVER STEP1: Create instances for socket and ServerSocket class.STEP2: Use the port 9000 for TCP.STEP3: Make the PrintStream object connect to the OuputStream using Socket.STEP4: Create an instance of the Date class and write it into the Socket.STEP5: Get the IP address of the client using the socket and getInetAddress ().STEP6: Print the client's IPAddress.
CLIENT STEP1: Create instances for socket class with the port number 9000.STEP2: Create an object of DataInputStream and make it to get data from server through the socket. STEP3: Read the Date object.STEP4: Print the obtained date.
Program:
Server:

```
import java.net.*;
import java.io.*;
class server4
{
public static void main(String[] args) throws Exception
{
```

```
ServerSocket s1 = new ServerSocket(8000);
System.out.println("Server Running");
Socket s2=s1.accept();
InetAddress a=InetAddress.getLocalHost();
String add=a.getHostAddress();
BufferedReader in=new BufferedReader(new
InputStreamReader(s2.getInputStream()));
PrintWriter out=new PrintWriter(s2.getOutputStream(),true);
System.out.println("Client Connected");
System.out.println(s2);
out.println(add);
System.out.println("Client's IP is ");
System.out.println(in.readLine());
s1.close();
s2.close();
}
}
```

Client:

```
import java.net.*;
import java.io.*;
class client4
{
public static void main(String[] args) throws Exception
{
InetAddress a=InetAddress.getLocalHost();
Socket s2 = new Socket(a,8000);
String add=a.getHostAddress();
BufferedReader in=new BufferedReader(new
InputStreamReader(s2.getInputStream()));
PrintWriter out=new PrintWriter(new
OutputStreamWriter(s2.getOutputStream()),true);
System.out.println(in.readLine());
out.println(add);
s2.close();
}
}
```

Aim: To perform a java program for UDP client and server.
Algorithm: SERVER: 1.Create a new Datagram Socket. 2.Create a new Datagram packet. 3.Create a message to be sent. 4.Convert into bytes. 5.create a packet. 6.send packet 7.wait for acknowledgement from client. 8.print data from client. 9.stop the program
CLIENT: 1. Create new Datagram Socket. 2.Create new Datagram packet. 3.Get the packet. 4.Print the content. 5.Create a new packet. 6.send to server. 7.Stop the program.
Server:

```
import java.net.*;
import java.io.*;
public class udpserver
{
public static int client=789;
public static int server=790;
public static void main(String arg[]) throws IOException
{
String s;
InetAddress id=InetAddress.getLocalHost();
BufferedReader dis=new BufferedReader(new InputStreamReader(System.in));
DatagramSocket ds=new DatagramSocket(server);
byte b[]=new byte[1024];
System.out.println("Server Side.... Sending....");
System.out.println("\n"+id);
while(true)
{
s=dis.readLine();
if(s.equals("end"))
{
b=s.getBytes();
DatagramPacket dp=new DatagramPacket(b,s.length(),id,client);
ds.send(dp);
break;
}
else
{
b=s.getBytes();
DatagramPacket dp=new DatagramPacket(b,s.length(),id,client);
ds.send(dp);
}
}
}
}
```

Client:

```
import java.net.*;
import java.io.*;
public class udpclient
{
public static int client=789;
public static void main(String args[]) throws IOException
{
DatagramSocket ds=new DatagramSocket(client);
byte b[]=new byte[1024];
System.out.println("client....receiving....");
while(true)
{
DatagramPacket dp=new DatagramPacket(b,b.length);
ds.receive(dp);
String s=new String(dp.getData(),0,dp.getLength());
if(s.equals("end")) break;
else System.out.println(s);
}
}
}
```

Aim: Program to create simple chat application.
Algorithm: SERVER STEP1: Instances of vector class is used to keep track of number of clients that can be connected and currently logged. STEP2: The method that is responsible for sending the message to the clients is made synchronized. STEP3: Server is capable of keeping into account the number of users. It adds and removes the client from the vector list as and when the connections are established and terminated.
CLIENT STEP1: The client receives the name of the user and message of that user and sends it to client. Server then passes it on to all clients connected.
Program:
Server:
```
import java.net.*;
import java.io.*;
import java.util.*;
public class cchatserver extends Thread
{
public static void main(String arg[])throws Exception
{
ServerSocket ssocket=new ServerSocket(4000);
Socket csocket=ssocket.accept();
BufferedReader br=new BufferedReader(new
InputStreamReader(csocket.getInputStream()));
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
PrintWriter out=new PrintWriter(csocket.getOutputStream(),true);
String s,t;
try
{
while(true)
{
System.out.println("server");
s=in.readLine();
out.println("server:"+s);
System.out.println(br.readLine());
}
}
catch(IOException e)
{
System.out.println("client has closed");
}
}
}
```
client:
```
import java.net.*;
import java.io.*;
public class cchatclient
{
public static void main(String arg[])throws Exception
{
Socket s=new Socket(InetAddress.getLocalHost(),4000);
BufferedReader br=new BufferedReader(new
InputStreamReader(s.getInputStream()));
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
PrintWriter out=new PrintWriter(s.getOutputStream(),true);
String input,t;
while(true)
{
System.out.println("client");
out.println("client:"+in.readLine());
System.out.println(br.readLine());
}
}
}
```

Aim: To create a Daemon program.
Algorithm:- Step 1: Create a Daemon class. Step 2:-Checking whether the thread is Daemon or not. Step 3: Setting user thread t1 to Daemon. Step 4:- Starting first 2 threads. Step 5:- Setting user thread t3 to Daemon.

```
 public class DaemonThread extends Thread
{
public DaemonThread(String name){
super(name);
}
public void run()
{
if(Thread.currentThread().isDaemon()){
System.out.println(getName() + " is Daemon thread");
}
else
{
System.out.println(getName() + " is User thread");
}
}
public static void main(String[] args)
{
DaemonThread t1 = new DaemonThread("t1");
DaemonThread t2 = new DaemonThread("t2");
DaemonThread t3 = new DaemonThread("t3");
t1.setDaemon(true);
t1.start();
t2.start();
t3.setDaemon(true);
t3.start();
}
}
```

Aim: Program to implement HTTP protocol and to print URl for the Client.
Algorithm:STEP 1: Create the URL with Http URL Connections.STEP 2: Define the Http Protocol for Client Connections.STEP3: Get the Http Connection.STEP4:Print the URL for the Client.
Program:

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

```
public class myhttp
{
public static void main(String args[])throws IOException
{
URL url=new URL("http://www.google.com/");
URLConnection conn=url.openConnection();
conn.connect();
InputStreamReader content= new InputStreamReader(conn.getInputStream());
FileWriter f=new FileWriter ("abc.html");
for(int i=0;i!=-1;i= content.read())
{
f.write((char) i);
}
}
}
```

Aim: Program to implement FTP using TCP.
Algorithm: CLIENT STEP 1: Create instance for the Socket class and establish connectivity with the server. STEP 2: Use the port number 4000. STEP 3: Receive the file from the server STEP 4: Reset the connection with the server
SERVER STEP 1: Create instances for the serversocket class and accept the server port. STEP 2: Read the filename to be opened. STEP 3: Send the file to the client
Program:
server:

```
import java.net.*;
import java.io.*;
public class ftpserver extends Thread
{
public static void main(String args[])
{
try
{
ServerSocket ss=new ServerSocket(4000);
Socket s=ss.accept();
BufferedReader br=new BufferedReader(new
InputStreamReader(s.getInputStream()));
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
PrintWriter out=new PrintWriter(s.getOutputStream(),true);
String fn,contents=" ",temp;
System.out.println("enter the file name to open: ");
fn=in.readLine();
File f=new File(fn);
if(f.isFile()&&f.canRead())
{
BufferedReader fil=new BufferedReader(new FileReader(fn));
while((temp=fil.readLine())!=null)
contents=contents+temp+"\n";
}
else
contents="error in input file";
System.out.println(" the contents of the file is:\n"+contents);
System.out.println("sending the file to the client....");
out.println(contents);
}
catch(Exception e)
{
System.out.println(e);
}
}
}
```

Client:

```
import java.net.*;
import java.io.*;
public class ftpclient {
public static void main(String args[])
{
try
{
Socket s=new Socket(InetAddress.getLocalHost(),4000);
BufferedReader br=new BufferedReader(new
InputStreamReader(s.getInputStream()));
String str;
while((str=br.readLine())!=null)
System.out.println(str);
}
catch(Exception e)
{
System.out.println("The connection to the server has been reset");
}
}
}
```

Aim: Program to implement FTP using UDP.
Algorithm: CLIENT STEP 1: Create instance for the Socket class and establish connectivity with the server. STEP 2: Use the port number 8000. STEP 3: Receive the file from the server STEP 4: Reset the connection with the server
SERVER STEP 1: Create instances for the serversocket class and accept the server port STEP 2: Read the filename to be opened. STEP 3: Send the file to the client
Program:
server:

```
import java.net.*;
import java.io.*;
public class Server
{
public static void main(String[] args) throws Exception
{
InetAddress sa=InetAddress.getByName(null);
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
String msg="",fn,tmp;
System.out.println("Enter the File name:");
fn=in.readLine();
File f=new File(fn);
if(f.isFile() && f.canRead())
{
BufferedReader fil=new BufferedReader(new FileReader(fn));
while((tmp=fil.readLine())!=null)
msg=msg+tmp+"\n";
}
else
msg="ERROR IN IMPUT FILE";
System.out.println(msg);
byte data[]=new byte[msg.length()];
msg.getBytes(0,msg.length(),data,0);
DatagramSocket ds = new DatagramSocket(8000);
DatagramPacket dp=new DatagramPacket(data,data.length,sa,8001);
ds.send(dp);
}
}
```

Client:

```
import java.net.*;
import java.io.*;
public class udpclient
{
public static void main(String[] args) throws Exception
{
InetAddress sa=InetAddress.getLocalHost();
byte data[]=new byte[1024];
DatagramSocket ds = new DatagramSocket(8001);
DatagramPacket dp=new DatagramPacket(data,data.length);
ds.receive(dp);
String msg=new String(dp.getData(),0,0,dp.getLength());
System.out.println("Received data : " + msg);
}
}
```

Aim:- Program to implement Traceroute command.
Algorithm:- Step 1: Import all the neccessary packages. Step 2:- Create the class trace route command. Step 3:- The get runtime command returns the runtine object associated with current java application. Step 4: Which website do you want to see the trace enter the Command due traces. Step 5: Stop the program.
Program:-

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class trace
{
 public static void runSystemCommand(String command)
 {
 try
 {
 Process p = Runtime.getRuntime().exec(command);
 BufferedReader inputStream = new BufferedReader(
 new InputStreamReader(p.getInputStream()));
 String s = "";
 while ((s = inputStream.readLine()) != null)
 System.out.println(s);
 }
 catch (Exception e)
 {
 }
 }
 public static void main(String[] args)
 {

 String ip = "www.youtube.com";
 runSystemCommand("tracert " + ip);
 }
}
```

Aim: Program to execute ping command.
Algorithm: Step 1: Import Neccessary packages. Step 2: Initized class ping. Step 3: Get Runtime exec command returns runtime object. Step 4: Buffered Reader function reader each byte line to line. Step 5: Print stark trace function used to handle exception and errors. Step 6:- Enter the required website for ping. Step 7:- Stop the program.

Program:

```
import java.io.*;
public class ping1
{
public static void runSystemCommand(String Command)
{
try{
Process p=Runtime.getRuntime().exec(Command);
BufferedReader InputStream=new BufferedReader(new InputStreamReader(p.getInputStream()));
String s="vvv";
while((s=InputStream.readLine())!=null)
{
System.out.println(s);
}
}
catch(Exception e)
{
e.printStackTrace();
}
}
public static void main(String[]args)
{
String Ip="localhost";
runSystemCommand("ping " +Ip);
java.util.Date date=new java.util.Date();
System.out.println(date);
}
}
```