**Program 1.Write a networking program in Java to implement a TCP server that provides services for a TCPClient.**

**TCP Client -**

|  |
| --- |
| import java.io.\*;  import java.net.\*; |
|  |
| class TCPClient{ |
| public static void main(String[] args){ |
| Socket client; |
|  |
| InputStreaminputStream; |
| DataInputStreamdataInputStream; |
|  |
| try{ |
| client = new Socket("localhost", 7313); |
| inputStream = client.getInputStream(); |
| dataInputStream =new DataInputStream(inputStream); |
| System.out.println(dataInputStream.readUTF()); |
| System.out.println(dataInputStream.readUTF()); |
| client.close(); |
| }catch(IOException e){ |
| System.out.println(e); |
| } |
|  |
| } |
| } |

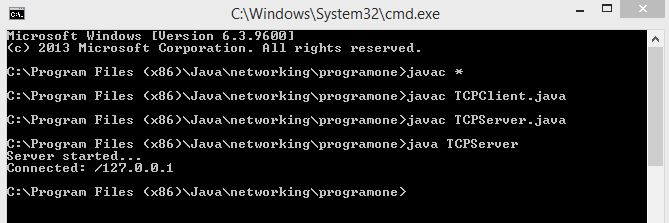
**TCP Server -**

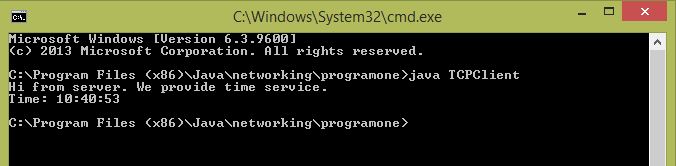
import java.io.\*;

import java.net.\*;

|  |
| --- |
| importjava.util.\*; |
|  |
|  |
| class TCPServer{ |
| public static void main(String[] args){ |
| ServerSocket server; |
| Socket client; |
|  |
| OutputStreamoutputStream; |
| DataOutputStreamdataOutputStream; |
|  |
| Calendar calendar; |
|  |
| try{ |
| server = new ServerSocket(7313); |
| System.out.println("Server started..."); |
| client = server.accept(); |
| System.out.println("Connected:" + client.getInetAddress()); |
|  |
| outputStream = client.getOutputStream(); |
| dataOutputStream = new DataOutputStream(outputStream); |
| dataOutputStream.writeUTF("Hi from server. We provide time service."); |
|  |
| calendar = Calendar.getInstance(); |
| dataOutputStream.writeUTF("Time:"+ calendar.get(Calendar.HOUR\_OF\_DAY) + ":" + calendar.get(Calendar.MINUTE) |
| + ":" + calendar.get(Calendar.SECOND)); |
| server.close(); |
| }catch(IOException e){ |
| System.out.println(e); |
| } |
| } |
| } |

Output:





**Program 2.Write a networking program to implement socketprogramming using User datagram Protocol inJava.**

**UDP Client –**

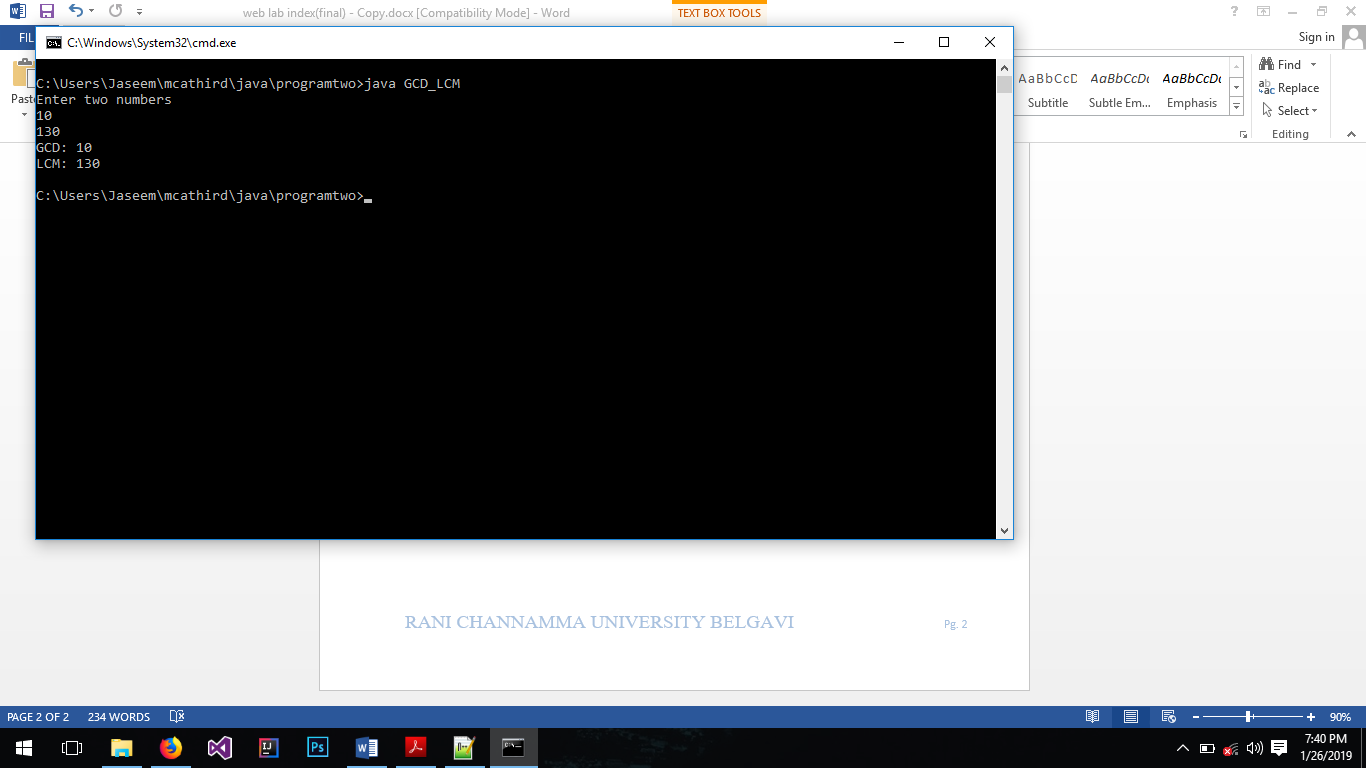
import java.net.\*;

import java.io.\*;

importjava.util.\*;

|  |
| --- |
| public class UDPClient { |
| public static void main(String[] args){ |
| DatagramSocketdatagramSocket; |
| DatagramPacketdatagramPacket; |
| String userInput; |
|  |
| InetAddressipAddr; |
| Scanner scanner = new Scanner(System.in); |
| byte[] bytes = new byte[1024]; |
|  |
| try{ |
| datagramSocket = new DatagramSocket(7314); |
| ipAddr = InetAddress.getByName("localhost"); |
| System.out.println("Write msg to send"); |
| userInput = scanner.nextLine(); |
| bytes = userInput.getBytes(); |
| datagramPacket =new DatagramPacket(bytes,bytes.length,ipAddr,7313); |
| datagramSocket.send(datagramPacket); |
| scanner.close(); |
| } catch (SocketException ex) { |
| System.out.println(ex); |
| } catch (UnknownHostException ex) { |
| System.out.println(ex); |
| } catch (IOException ex) { |
| System.out.println(ex); |
| } |
|  |
| } |
| } |

Output:



**UDP Server –**

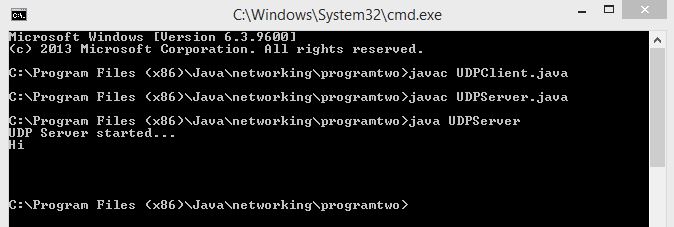
import java.net.\*;

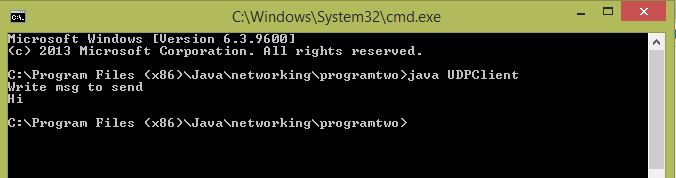
import java.io.\*;

importjava.util.\*;

|  |
| --- |
| public static void main(String[] args) { |
| DatagramSocketdatagramSocket; |
| DatagramPacketdatagramPacket; |
| byte[] bytes; |
| String msg; |
|  |
| try { |
| datagramSocket = new DatagramSocket(7313); |
| System.out.println("UDP Server started..."); |
| bytes = new byte[1024]; |
| datagramPacket = new DatagramPacket(bytes, 0, bytes.length); |
| datagramSocket.receive(datagramPacket); |
| msg = new String(bytes); |
| System.out.println(msg); |
| } catch (SocketException ex) { |
| System.out.println(ex); |
| } catch (IOException ex) { |
| System.out.println(ex); |
| } |
|  |
| } |
| } |

Output:



****

**Program 3.Implement an FTP server using socketprogramming.**

**FTP Client -**

import java.net.\*;

import java.io.\*;

class FTPClient{

public static void main(String[] args){

String fileName;

File file;

FileOutputStream fileOutputStream = null;

InputStream inputStream = null;

DataInputStream dataInputStream = null;

Socket socket = null;

byte[] byteArray;

try {

socket = new Socket("localhost", 7313);

System.out.println("Connected...");

inputStream = socket.getInputStream();

dataInputStream = new DataInputStream(inputStream);

fileName = dataInputStream.readUTF();

file = new File(fileName);

fileOutputStream = new FileOutputStream(file);

int bytesRead = 0;

byteArray = new byte[1024 \* 10];

while((bytesRead = inputStream.read(byteArray)) > -1){

fileOutputStream.write(byteArray, 0, bytesRead);

}

System.out.println("Received Successfully...");

fileOutputStream.close();

socket.close();

} catch (IOException e) {

System.err.println("Error 1");

e.printStackTrace();

}

}

}

Output:

**FTPServer -**

import java.net.\*;

import java.io.\*;

import java.util.\*;

class FTPServer{

public static void main(String[] args){

String fileName = null;

FileInputStream fileInputStream = null;

OutputStream outputStream = null;

DataOutputStream dataOutputStream = null;

ServerSocket serverSocket = null;

Socket client = null;

Scanner scan = new Scanner(System.in);

File file;

byte[] bytes;

try

{

serverSocket = new ServerSocket(7313);

System.out.println("FTP Server started...");

client = serverSocket.accept();

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

outputStream = client.getOutputStream();

dataOutputStream = new DataOutputStream(outputStream);

System.out.println("Enter file name to send");

fileName = scan.nextLine();

file = new File(fileName);

fileInputStream = new FileInputStream(file);

int fileLength = (int) file.length();

bytes = new byte[fileLength];

fileInputStream.read(bytes, 0, bytes.length);

dataOutputStream.writeUTF(file.getName());

outputStream.write(bytes, 0, bytes.length);

outputStream.close();

scan.close();

System.out.println("Sent Successfully");

serverSocket.close();

} catch (IOException e) {

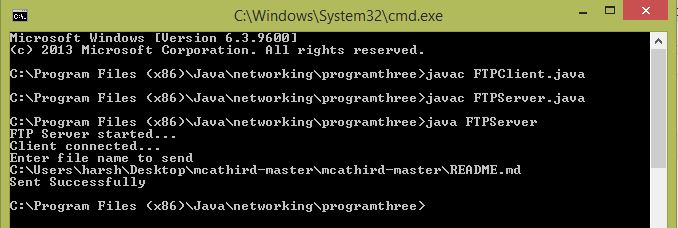
e.printStackTrace();

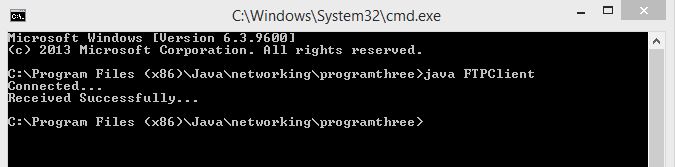
}

}

}

Output:





**Program 4.Implement a chat server using socketprogramming.**

**Chat Client -**

import java.net.\*;

import java.io.\*;

import java.util.\*;

class ChatClient{

public static void main(String[] args){

Socket socket = null;

Scanner scan = new Scanner(System.in);

String yourMsg, serverMsg;

InputStream inputStream = null;

OutputStream outputStream = null;

DataInputStream dataInputStream = null;

DataOutputStream dataOutputStream = null;

try{

socket = new Socket("localhost", 7313);

System.out.println("Connected...");

inputStream = socket.getInputStream();

outputStream = socket.getOutputStream();

dataInputStream = new DataInputStream(inputStream);

dataOutputStream = new DataOutputStream(outputStream);

while(true){

serverMsg = dataInputStream.readUTF();

if(serverMsg.equals("exit")){

break;

}

System.out.println("Server: " + serverMsg);

System.out.print("You: ");

yourMsg = scan.nextLine();

dataOutputStream.writeUTF(yourMsg);

if(yourMsg.equals("exit")){

break;

}

}

socket.close();

}catch(IOException e){

System.out.println(e);

}

}

}

**ChatServer -**

import java.net.\*;

import java.io.\*;

import java.util.\*;

class ChatServer{

public static void main(String[] args){

ServerSocket serverSocket = null;

Socket client = null;

Scanner scan = new Scanner(System.in);

String yourMsg, clientMsg;

OutputStream outputStream = null;

InputStream inputStream = null;

DataOutputStream dataOutputStream = null;

DataInputStream dataInputStream = null;

try {

serverSocket = new ServerSocket(7313);

System.out.println("Server started...");

client = serverSocket.accept();

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

outputStream = client.getOutputStream();

inputStream = client.getInputStream();

dataOutputStream = new DataOutputStream(outputStream);

dataInputStream = new DataInputStream(inputStream);

dataOutputStream.writeUTF("Hi from server");

while(true){

clientMsg = dataInputStream.readUTF();

if(clientMsg.equals("exit")){

break;

}

System.out.println("Client: " + clientMsg);

System.out.print("You: ");

yourMsg = scan.nextLine();

dataOutputStream.writeUTF(yourMsg);

if(yourMsg.equals("exit")){

break;

}

}

serverSocket.close();

}catch(IOException e){

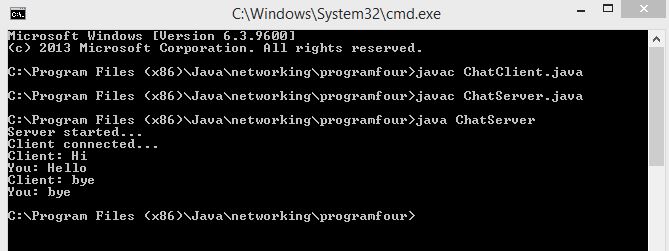
System.out.println(e);

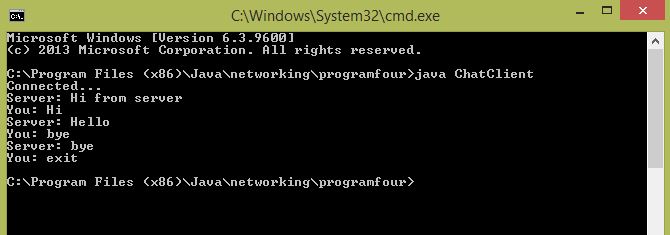
}

}

}

Output:





**Program 5.Implement an ECHO server using socketprogramming.**

**ECHO Client -**

import java.net.\*;

import java.io.\*;

import java.util.\*;

class Client{

public static void main(String[] args){

Socket socket = null;

Scanner scan = new Scanner(System.in);

String yourMsg, echo;

InputStream inputStream = null;

OutputStream outputStream = null;

DataInputStream dataInputStream = null;

DataOutputStream dataOutputStream = null;

try{

socket = new Socket("localhost", 7313);

System.out.println("Connected...");

inputStream = socket.getInputStream();

outputStream = socket.getOutputStream();

dataInputStream = new DataInputStream(inputStream);

dataOutputStream = new DataOutputStream(outputStream);

System.out.println("Enter msg to echo...");

yourMsg = scan.nextLine();

dataOutputStream.writeUTF(yourMsg);

echo = dataInputStream.readUTF();

System.out.println(echo);

System.out.println("Echoed successfully");

socket.close();

scan.close();

}catch(IOException e){

System.out.println(e);

}

}

}

**ECHOServer -**

import java.net.\*;

import java.io.\*;

class EchoServer{

public static void main(String[] args){

ServerSocket serverSocket = null;

Socket client = null;

String clientMsg;

InputStream inputStream = null;

OutputStream outputStream = null;

DataInputStream dataInputStream = null;

DataOutputStream dataOutputStream = null;

try

{

serverSocket = new ServerSocket(7313);

System.out.println("Server started...");

client = serverSocket.accept();

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

inputStream = client.getInputStream();

outputStream = client.getOutputStream();

dataInputStream = new DataInputStream(inputStream);

dataOutputStream = new DataOutputStream(outputStream);

clientMsg = dataInputStream.readUTF();

System.out.println("Echoing...");

dataOutputStream.writeUTF(clientMsg);

serverSocket.close();

}catch(IOException e){

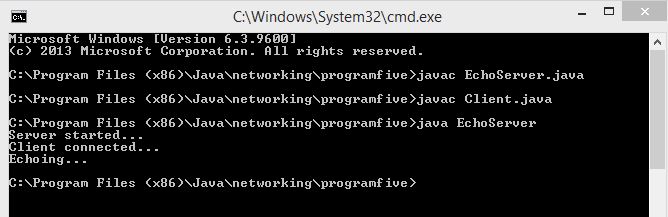
System.out.println(e);

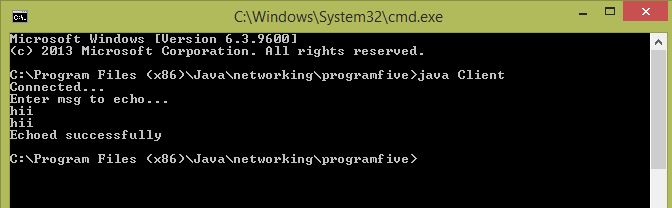
}

}

}

Output:





**Program 6.Implement Address Resolution Protocol using socketprogramming.**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class ARPDemo {

public static void main(String[] args) {

String ip;

Scanner scan = new Scanner(System.in);

ProcessBuilder processBuilder = new ProcessBuilder();

Process process;

System.out.println("Enter the ip address");

ip = scan.nextLine();

InputStream is;

try{

InetAddress inet = InetAddress.getByName(ip);

if(inet.isReachable(5000)){

process = processBuilder.command("arp", "-a").start();

is = process.getInputStream();

BufferedReader buff = new BufferedReader(new InputStreamReader(is));

String res;

while((res = buff.readLine()) != null){

if(res.contains(ip)){

res = res.trim();

res = res.replaceAll(" +", " ");

String[] array = res.split(" ");

System.out.println(array[0] + " ==> " + array[1]);

}

}

}else{

System.out.println("Host is not present");

}

}catch(Exception e){

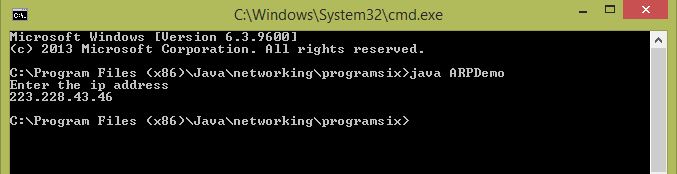
System.out.println(e);

}

}

}

Output:



**Program 7.Implement Ping server and Ping client using socketprogramming.**

**Ping Client –**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class PingClient {

public static void main(String[] args){

DatagramSocket socket;

DatagramPacket packet;

InetAddress ipaddr;

byte[] bytes;

int n = 0;

String str;

long sTime;

long rTime;

long cTime;

try{

socket = new DatagramSocket(7312);

ipaddr = InetAddress.getByName("localhost");

while(n < 5){

str = "dummy packet";

bytes = str.getBytes();

packet=new DatagramPacket(bytes,bytes.length,ipaddr,7313);

socket.send(packet);

sTime = new Date().getTime();

try{

byte[] rBytes = new byte[1024];

DatagramPacket rPacket = new DatagramPacket(rBytes, 0, rBytes.length);

socket.setSoTimeout(5000); socket.receive(rPacket);

rTime = new Date().getTime();

cTime = rTime - sTime;

System.out.println("Reply from " + rPacket.getAddress().toString() + ": time < " +

cTime + "ms");

}catch(IOException ex){

System.out.println("Request Timeout: " + n);

}

Thread.sleep(1000);

n++;

}

} catch (SocketException ex) {

System.out.println(ex.getMessage());

} catch (UnknownHostException ex) {

System.out.println(ex.getMessage());

} catch (IOException ex) {

System.out.println(ex.getMessage());

} catch (InterruptedException ex) {

System.out.println(ex.getMessage());

}

}

}

**PingServer –**

import java.net.\*;

import java.io.\*;

public class PingServer {

public static void main(String[] args){

DatagramSocket sock;

DatagramPacket packet;

byte[] rBytes = new byte[1024];

String msg;

String str;

int n = 0;

try {

sock = new DatagramSocket(7313);

System.out.println("Server started");

packet = new DatagramPacket(rBytes, 0, rBytes.length);

while(n < 5){

sock.receive(packet);

msg = new String(rBytes);

System.out.println(packet.getAddress().toString() + ": is Pinging");

byte[] sBytes;

str = "dummy packet";

sBytes = str.getBytes();

DatagramPacket sPacket = new DatagramPacket(sBytes,0, sBytes.length, InetAddress.getByName("localhost"), 7312);

sock.send(sPacket);

n++;

}

} catch (SocketException ex) {

System.out.println(ex.getMessage());

} catch (IOException ex) {

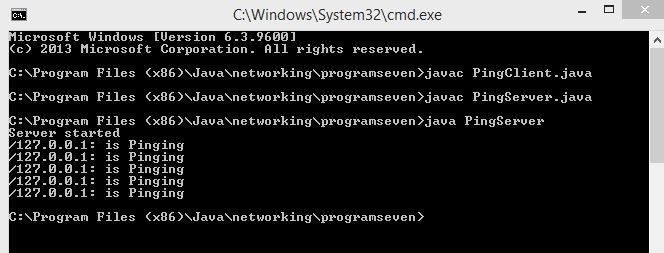
System.out.println(ex.getMessage());

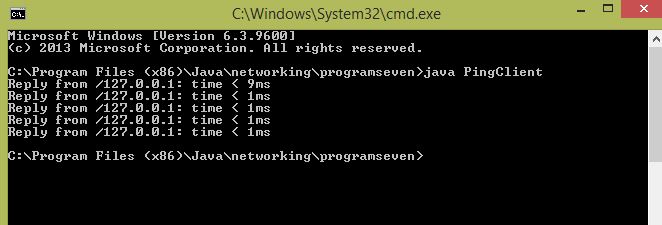
}

}

}

Output:

****



**Program 8.Implement Remote Command Execution using networkprogramming.**

**RCE Client –**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class RCEClient {

public static void main(String[] args){

Socket client;

InputStream is;

OutputStream os;

DataOutputStream dos;

DataInputStream dis;

Scanner scan = new Scanner(System.in);

String cmd;

try{

client = new Socket("localhost", 7313);

System.out.println("Connected to server: " + client.getInetAddress());

System.out.println("Enter the command to execute remotely");

cmd = scan.nextLine();

is = client.getInputStream();

os = client.getOutputStream();

dis = new DataInputStream(is);

dos = new DataOutputStream(os);

dos.writeUTF(cmd);

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

os.flush();

client.close();

}catch(IOException e){

System.out.println(e.getMessage());

}

}

}

**RCEServer –**

import java.net.\*;

import java.io.\*;

class RCEServer {

public static void main(String[] args) {

ServerSocket server;

Socket client;

OutputStream os;

InputStream is;

InputStream commandInputStream;

DataInputStream dis;

DataOutputStream dos;

String[] command;

Process process;

ProcessBuilder pB = new ProcessBuilder();

try{

server = new ServerSocket(7313);

System.out.println("Server started...");

client = server.accept();

System.out.println("Client connected: " + client.getInetAddress());

os = client.getOutputStream();

is = client.getInputStream();

dis = new DataInputStream(is);

dos = new DataOutputStream(os);

String cmd = dis.readUTF();

cmd = cmd.trim();

command = cmd.split(" ");

process = pB.command(command).start();

commandInputStream = process.getInputStream();

BufferedReader buff = new BufferedReader(new InputStreamReader(commandInputStream));

String read;

String result = "";

while ((read = buff.readLine()) != null){

result = read + "\n" + result;

}

dos.writeUTF(result);

dos.flush();

System.out.println("Command: " + cmd + " successfully.");

System.out.println("Server exited");

server.close();

}catch(IOException e){

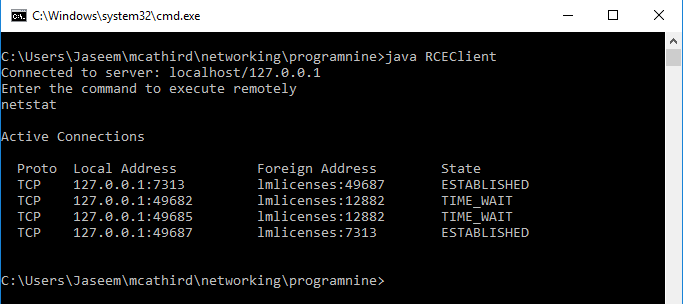
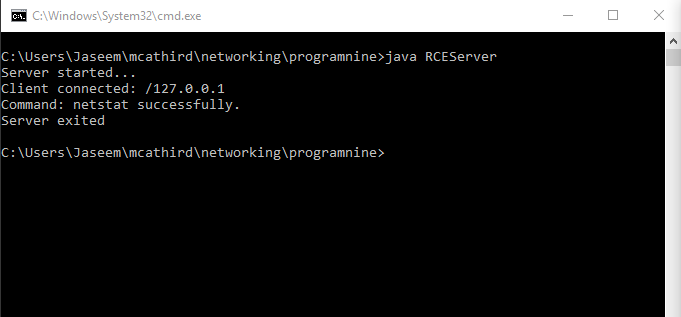
System.out.println(e.getMessage());

}

}

}

Output:



**Program 9.Implement a program to retrieve the data for the specifiedURL.**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class RetrieveData {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

HttpURLConnection connection;

URL url;

String input;

try {

System.out.println("Enter the URL");

input = scan.nextLine();

url = new URL(input);

connection = (HttpURLConnection) url.openConnection();

System.out.println("Request Method: " + connection.getRequestMethod());

System.out.println("Response Code: " + connection.getResponseCode());

System.out.println("Response Message: " + connection.getResponseMessage());

Map<String, List<String>> headerFields = connection.getHeaderFields();

Set<String> headerKeys = headerFields.keySet();

for(String key: headerKeys){

System.out.println("Key: " + key + " : " + "Value: " + headerFields.get(key));

}

connection.disconnect();

scan.close();

} catch (Exception ex) {

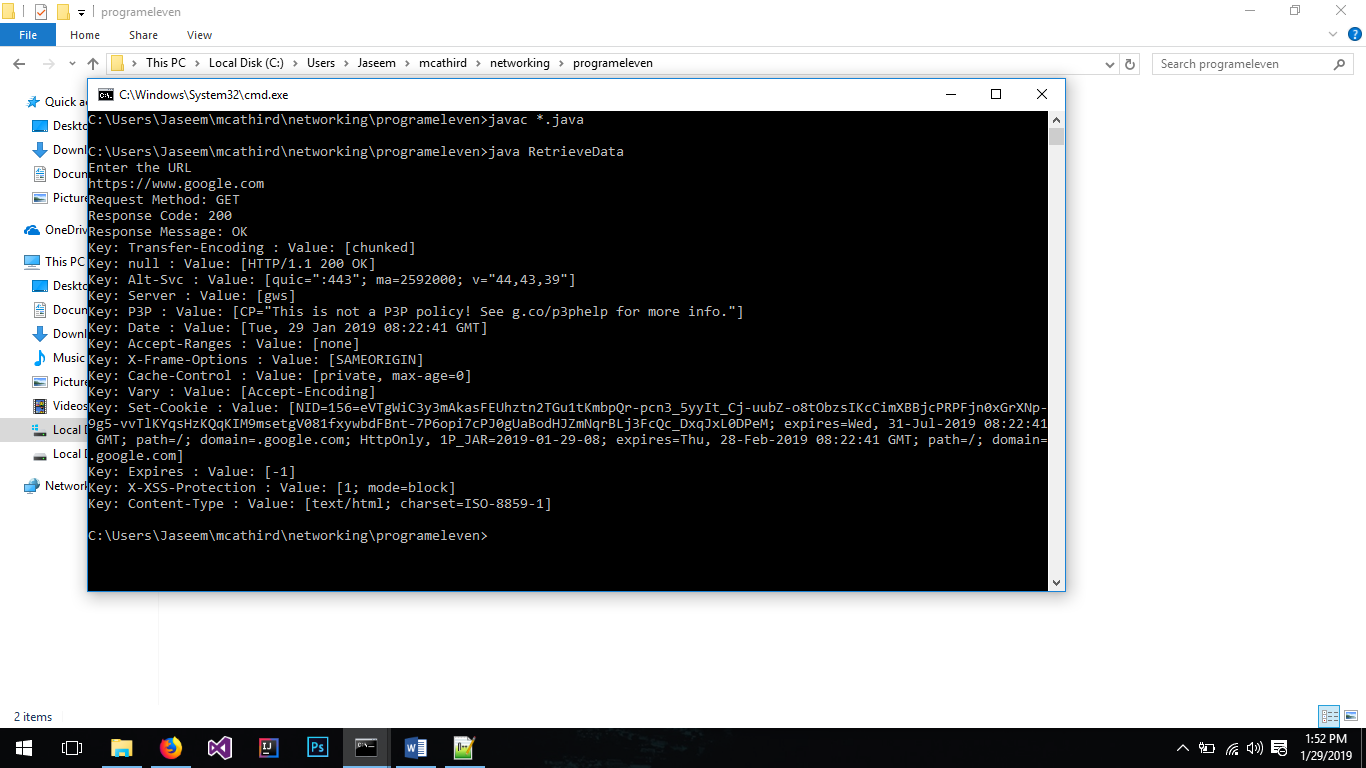
System.out.println(ex);

}

}

}

Output:



**Program 10.Write a Java program to check whether the given DNS is found in the internet ornot.**

import java.net.\*;

import java.util.\*;

class DNSTest{

public static void main(String[] args){

String host = new String();

Scanner input = new Scanner(System.in);

InetAddress inetAddress;

try{

System.out.println("Enter host name");

host = input.nextLine();

inetAddress = InetAddress.getByName(host);

System.out.println("Host Name: " + inetAddress.getHostName());

System.out.println("Host Address: " + inetAddress.getHostAddress());

System.out.println();

}catch(UnknownHostException e){

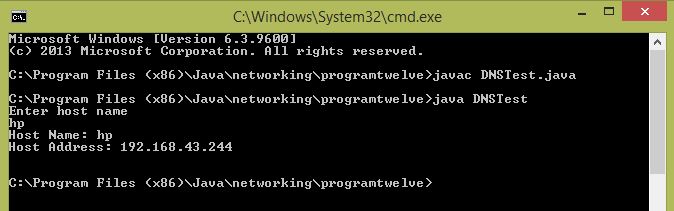
System.out.println("Host not found: " + host);

}

}

}

Output:



**Program 11.Write a network program using HTTP to print the document for the givenURL.**

import java.net.\*;

import java.io.\*;

import java.util.\*;

public class HttpDocumentPrinter {

public static void main(String[] args) {

Scanner scan = new Scanner(System.in);

HttpURLConnection connection;

URL url;

InputStream inputStream;

String input;

try {

System.out.println("Enter the URL");

input = scan.nextLine();

url = new URL(input);

connection = (HttpURLConnection) url.openConnection();

inputStream = connection.getInputStream();

int read;

while((read = inputStream.read()) > -1){

char ch = (char) read;

System.out.print(ch);

}

scan.close();

} catch (Exception ex) {

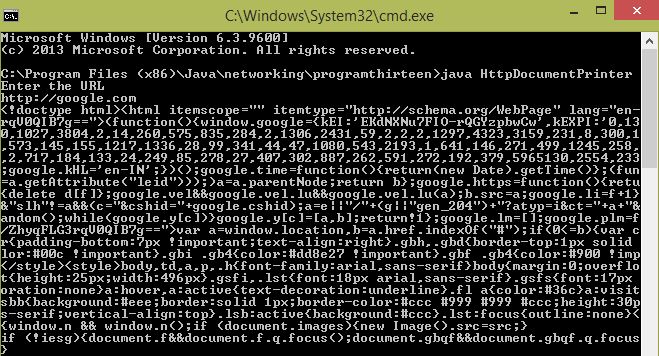
System.out.println(ex);

}

}

}

Output:



**Program 12.Implementation of STAR topology.**

set ns [new Simulator]

set f [open "Out.tr" w]

$ns trace-all $f

setfr [open "out.nam" w]

$ns namtrace-all $fr

proc finish {} {

global ns f fr

$ns flush-trace

close $f

close $fr

execnamout.nam&

exit

}

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$ns duplex-link $n1 $n0 2Mb 5ms DropTail

$ns duplex-link $n2 $n0 2Mb 5ms DropTail

$ns duplex-link $n3 $n0 2Mb 5ms DropTail

$ns duplex-link $n4 $n0 2Mb 5ms DropTail

$ns duplex-link $n5 $n0 2Mb 5ms DropTail

set tcp0 [new Agent/TCP]

$ns attach-agent $n1 $tcp0

set ftp [new Application/FTP]

$ftp attach-agent $tcp0

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp0 $sink

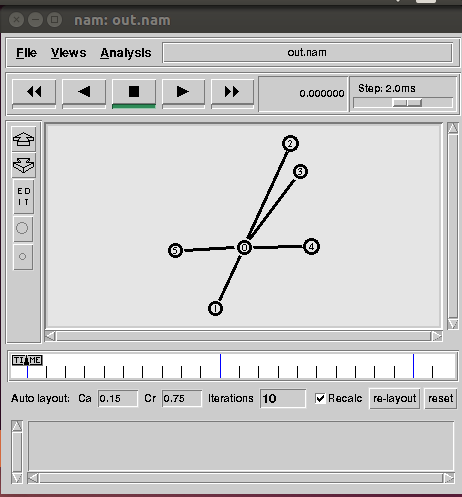
$ns at .1 "$ftp start"

$ns at 2 "$ftp stop"

$ns at 2.1 "finish"

$ns run

Output:



**Program 13.Monitoring traffic for the given topology.**

set ns [new Simulator]

#Open the nam trace file

setnf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish {} {

global ns nf

$ns flush-trace

#Close the trace file

close $nf

#Executenam on the trace file

execnamout.nam&

exit 0

}

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

#CreateLanbetween the nodes

set lan0 [$ns newLan "$n0 $n1 $n2 $n3 $n4" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

#Connect the traffic sources with the traffic sink

$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize\_ 500

$cbr0 set interval\_ 0.01

$cbr0 attach-agent $tcp0

#Schedule events for the CBR agents

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run

Output:

