1: Write a program for error detecting code using CRC-CCITT (16-bits).

import java.io.\*;

import java.lang.\*;

import java.util.\*;

class Main

{

public static String string\_val(String sts,int poly\_length)

{

for(int i=1;i<poly\_length;i++)

{

sts=sts+"0";

}

return sts;

}

public static String generate(char[] divisior,char[] dividend,int len,String org)

{

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

{

if(dividend[i]=='1')

{

for(int j=0;j<divisior.length;j++)

{

if(dividend[i+j]==divisior[j])

{

dividend[i+j]='0';

}

else

{

dividend[i+j]='1';

}

}

}

}

String st=String.valueOf(dividend);

String fin=org+st.substring(len);

return fin;

}

public static void main(String[] args)

{

String str,rec;

String d="10001000000100001";

Scanner sc=new Scanner(System.in);

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

str=sc.next();

String org=str;

// System.out.println("ORIGINAL STRING IS " + org);

int len=str.length();

str=string\_val(str,d.length());

char[] divisior=d.toCharArray();

char[] dividend=str.toCharArray();

String fin=generate(divisior,dividend,len,org);

System.out.println("DIVISIOR= " + String.valueOf(divisior));

System.out.println("DIVIDEND= " + String.valueOf(dividend));

System.out.println("TRANSMITTED MESSAGE IS " + fin);

System.out.println("Enter the received message");

rec=sc.next();

org=rec;

len=rec.length();

rec=string\_val(rec,d.length());

dividend=rec.toCharArray();

String rin=generate(divisior,dividend,len,org);

System.out.println("MESSAGE DUE TO ERRORS IS");

System.out.println(rin);

if(fin.equals(rin))

{

System.out.println("NO ERRORS");

}

else

{

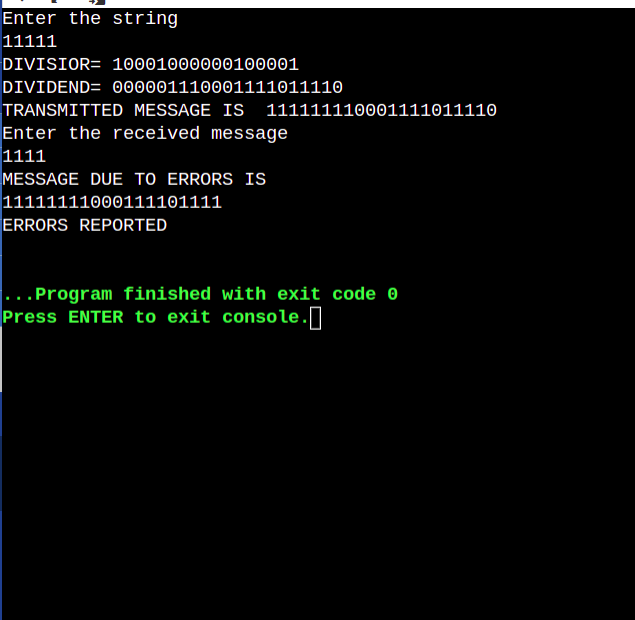
System.out.println("ERRORS REPORTED");

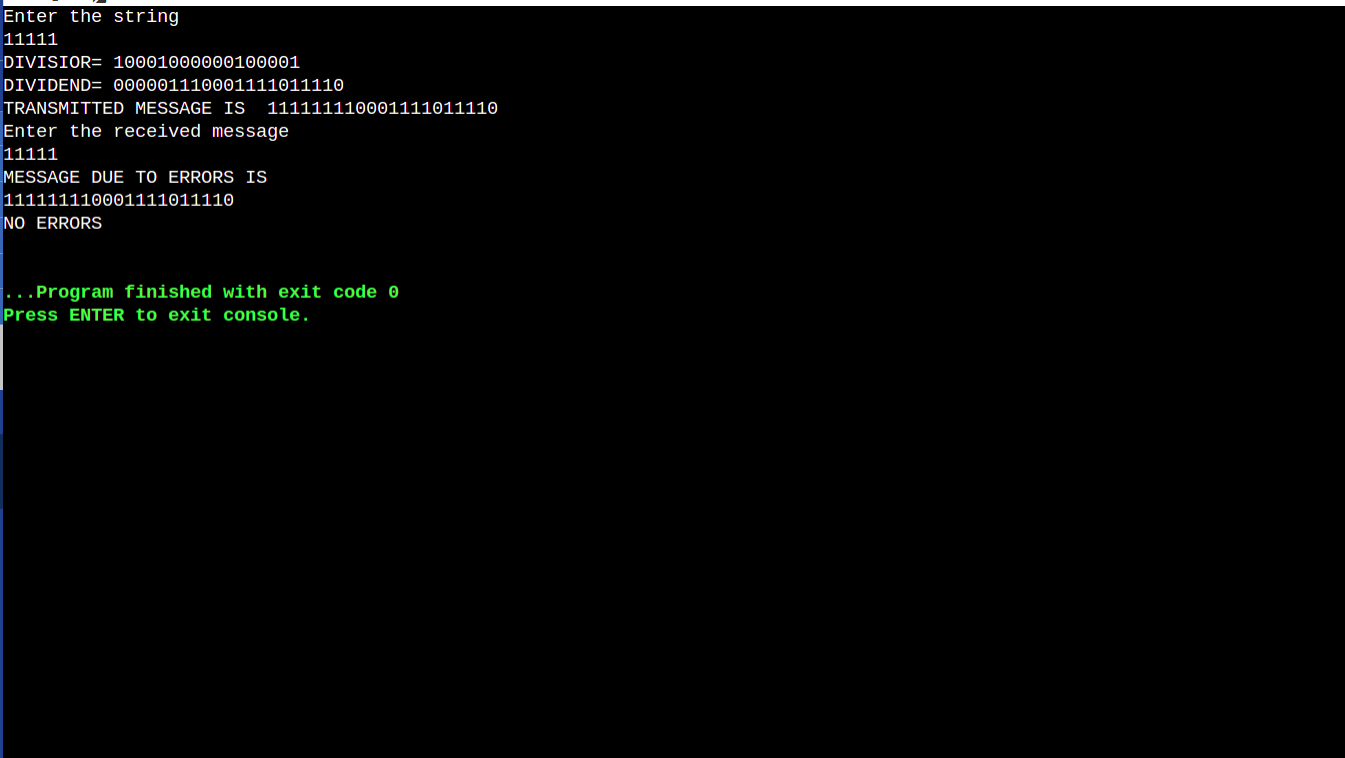
}

// System.out.println

}

}





2: Write a program for distance vector algorithm to find suitable path for

transmission.

#include<stdio.h>

struct node

{

unsigned dist[20];

unsigned from[20];

}rt[10];

int main()

{

int dmat[20][20];

int n,i,j,k,count=0;

printf("\nEnter the number of nodes : ");

scanf("%d",&n);

printf("\nEnter the cost matrix :\n");

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

for(j=0;j<n;j++)

{

scanf("%d",&dmat[i][j]);

dmat[i][i]=0;

rt[i].dist[j]=dmat[i][j];

rt[i].from[j]=j;

}

do

{

count=0;

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

for(j=0;j<n;j++)

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

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

{

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<n;i++)

{

printf("\n\nState value for router %d is \n",i+1);

for(j=0;j<n;j++)

{

printf(" \t\nnode %d via %d Distance%d ",j+1,rt[i].from[j]+1,rt[i].dist[j]);

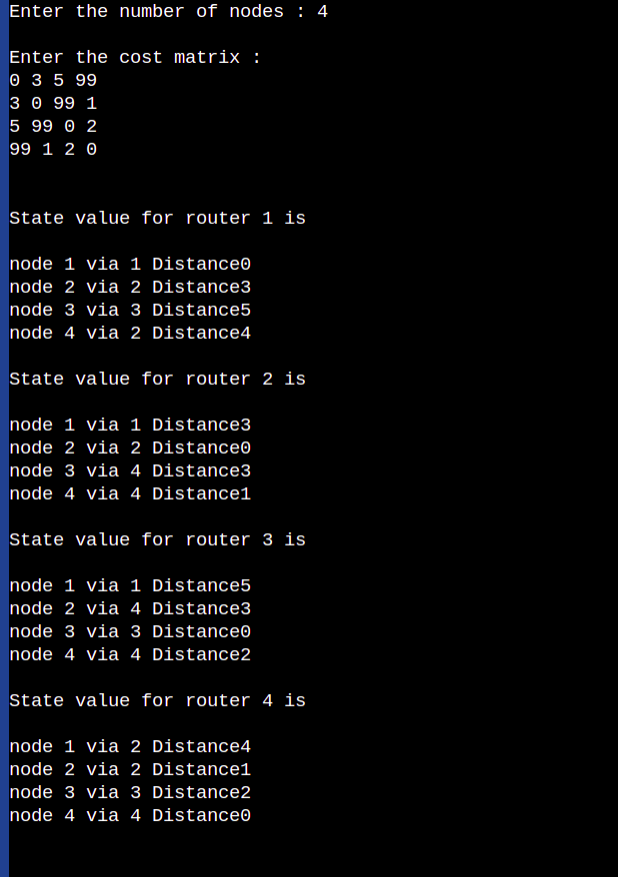
}

}

printf("\n\n");

}

OUTPUT:



3: Implement Dijkstra’s algorithm to compute the shortest path for a given

topology.

import java.io.\*;

import java.lang.\*;

import java.util.\*;

public class djksitra

{

public int graph[][];

int src;

int n;

public djksitra(int n,int src)

{

this.n=n;

graph=new int[n][n];

this.src=src;

}

public void input()

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the Distance matrix");

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

{

for(int j=0;j<n;j++)

{

graph[i][j]=sc.nextInt();

}

}

}

public int mindistance(int dist[],Boolean val[])

{

int min=Integer.MAX\_VALUE;

int min\_index=-1;

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

{

if(dist[i]<min && val[i]==false)

{

min=dist[i];

min\_index=i;

}

}

return min\_index;

}

public int[] dj()

{

int dist[]=new int[n];

Boolean val[]=new Boolean[n];

Arrays.fill(dist,Integer.MAX\_VALUE);

Arrays.fill(val,false);

dist[src]=0;

for(int i=0;i<n-1;i++)

{

int u=mindistance(dist,val);

val[u]=true;

for(int j=0;j<n;j++)

{

if(val[j]==false && graph[u][j]!=0 && dist[u] + graph[u][j]<dist[j])

{

dist[j]=dist[u] + graph[u][j];

}

}

}

return dist;

}

}

MAIN.JAVA

import java.io.\*;

import java.util.\*;

class Main

{

public static void main(String[] args)

{

int n;

int src;

Scanner sc=new Scanner(System.in);

System.out.println("Enter the Number of nodes");

n=sc.nextInt();

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

src=sc.nextInt();

djksitra ob=new djksitra(n,src);

ob.input();

int[] ans=ob.dj();

System.out.println("SHORTEST DISTANCE FROM THE SOURCE " + src);

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

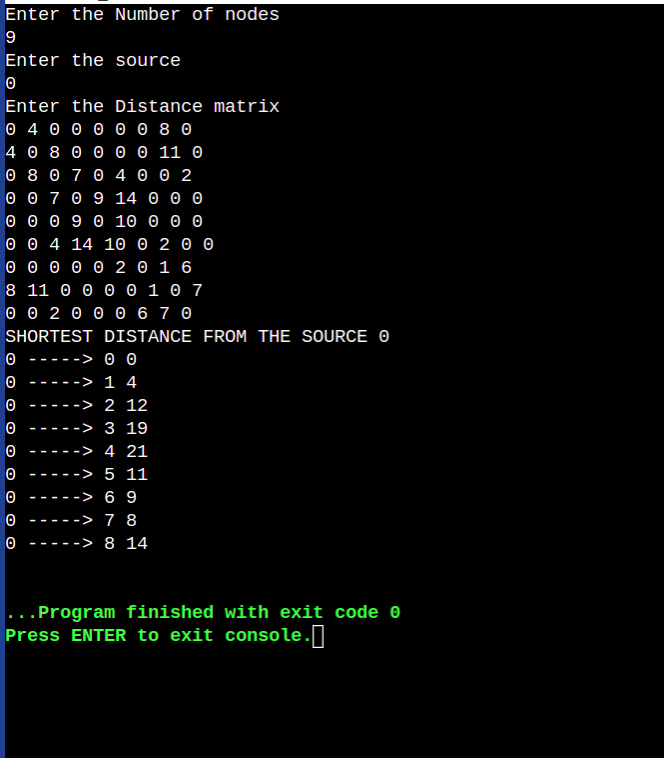
{

System.out.println(src + " -----> " + i + " " + ans[i]);

}

}

}



4:Write a program for congestion control using Leaky bucket algorithm.

import java.io.\*;

import java.lang.\*;

import java.util.\*;

public class leakey\_bucket

{

public int no\_of\_packets;

public int bucket\_size;

public int o\_rate;

public int packet\_size\_remaining;

public int op;

public int packet\_size[];

leakey\_bucket(int size)

{

no\_of\_packets=size;

bucket\_size=0;

o\_rate=0;

packet\_size\_remaining=0;

op=0;

packet\_size=new int[size];

}

public void input()

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the bucket size");

bucket\_size=sc.nextInt();

System.out.println("Enter the Output Rate");

o\_rate=sc.nextInt();

System.out.println("Enter the size of the each of the packets");

for(int i=0;i<no\_of\_packets;i++)

{

packet\_size[i]=sc.nextInt();

}

}

public void calc()

{

for(int i=0;i<no\_of\_packets;i++)

{

if(packet\_size[i] + packet\_size\_remaining > bucket\_size)

{

if(packet\_size[i]>bucket\_size)

{

System.out.println("INCOMING PACKET SIZE " + packet\_size[i] + " GREATER THAN BUCKET CAPACITY " + bucket\_size);

System.out.println();

}

else

{

System.out.println("BUCKET CAPACITY EXCEEDED PACKET\_REJECTED");

}

}

else

{

packet\_size\_remaining+=packet\_size[i];

System.out.println("INCOMING PACKET SIZE " + packet\_size[i]);

System.out.println("BYTE REMAINING TO BE TRANSMITTED " + packet\_size\_remaining);

System.out.println();

while(packet\_size\_remaining>0)

{

if(packet\_size\_remaining>0)

{

if(packet\_size\_remaining<=o\_rate)

{

op=packet\_size\_remaining;

packet\_size\_remaining=0;

}

else

{

op=o\_rate;

packet\_size\_remaining-=o\_rate;

}

System.out.println("PACKET SIZE TRANSMITTED " + op);

System.out.println("BYTES REMAINING " + packet\_size\_remaining);

System.out.println();

}

else

{

System.out.println("NO PACKETS TO TRANSMIT " );

}

}

}

}

}

}

MAIN.JAVA

import java.io.\*;

import java.util.\*;

public class Main

{

public static void main(String [] args)

{

Scanner sc=new Scanner(System.in);

int n;

System.out.println("Enter the number of packets");

n=sc.nextInt();

leakey\_bucket ob=new leakey\_bucket(n);

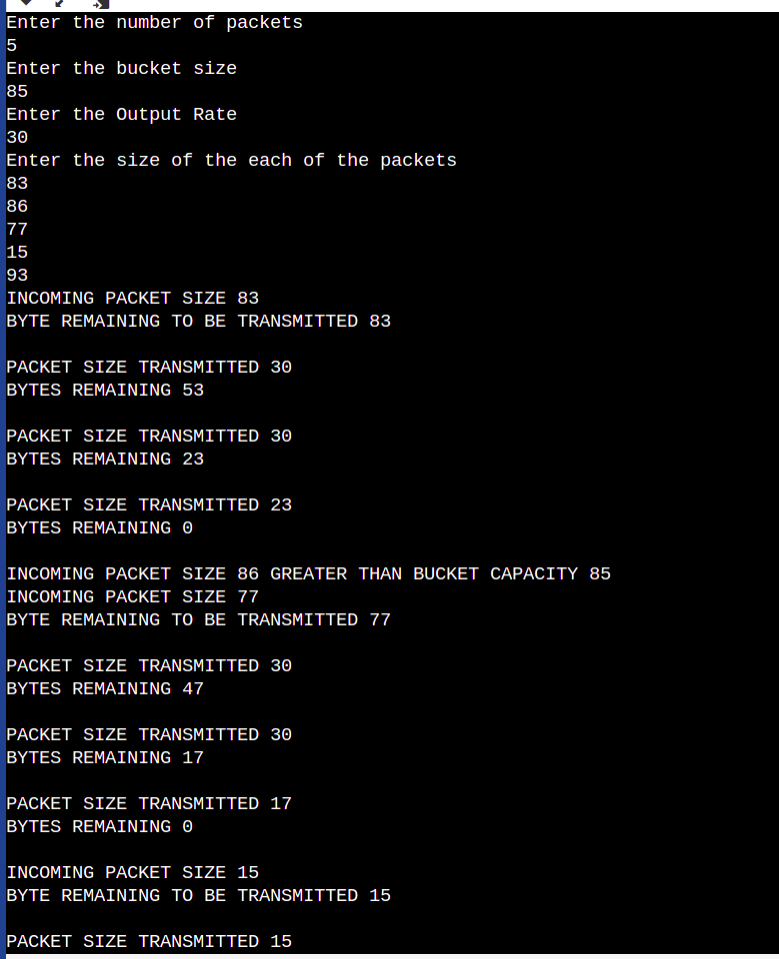
ob.input();

ob.calc();

}

}

Output:



5:Using TCP/IP sockets, write a client-server program to make client sending

the file name and the server to send back the contents of the requested file if

present.

import java.io.\*;

import java.net.\*;

import java.lang.\*;

import java.util.\*;

public class tcpclient

{

public static void main(String[] args)

{

try

{

Scanner sc=new Scanner(System.in);

Socket s=new Socket("127.0.0.1",12000);

DataOutputStream dos=new DataOutputStream(s.getOutputStream());

DataInputStream dis=new DataInputStream(s.getInputStream());

dos.writeUTF("CONNECTED TO 127.0.0.1\n");

System.out.println("Enter the path of the file");

String path=sc.nextLine();

dos.writeUTF(path);

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

dis.close();

dos.close();

sc.close();

s.close();

}

catch(Exception e)

{

System.out.println(e);

}

}

}

import java.io.\*;

import java.net.\*;

import java.lang.\*;

import java.util.\*;

public class tcpserver

{

public static void main(String[] args)

{

try

{

ServerSocket s=new ServerSocket(12000);

System.out.println("SERVER WAITING FOR CONNECTION ..... ");

Socket s1=s.accept();

DataOutputStream dos=new DataOutputStream(s1.getOutputStream());

DataInputStream dis=new DataInputStream(s1.getInputStream());

System.out.println("STATUS: " + dis.readUTF());

String path=dis.readUTF();

System.out.println("REQUEST HAS BEEN RECEIVED ");

try

{

File mf=new File(path);

Scanner sc=new Scanner(mf);

String st=sc.nextLine();

while(sc.hasNextLine())

{

st=st+"\n" + sc.nextLine();

}

dos.writeUTF(st);

dos.close();

s1.close();

s.close();

sc.close();

}

catch(FileNotFoundException e)

{

dos.writeUTF("FILE NOT FOUND ");

}

}

catch(IOException e)

{

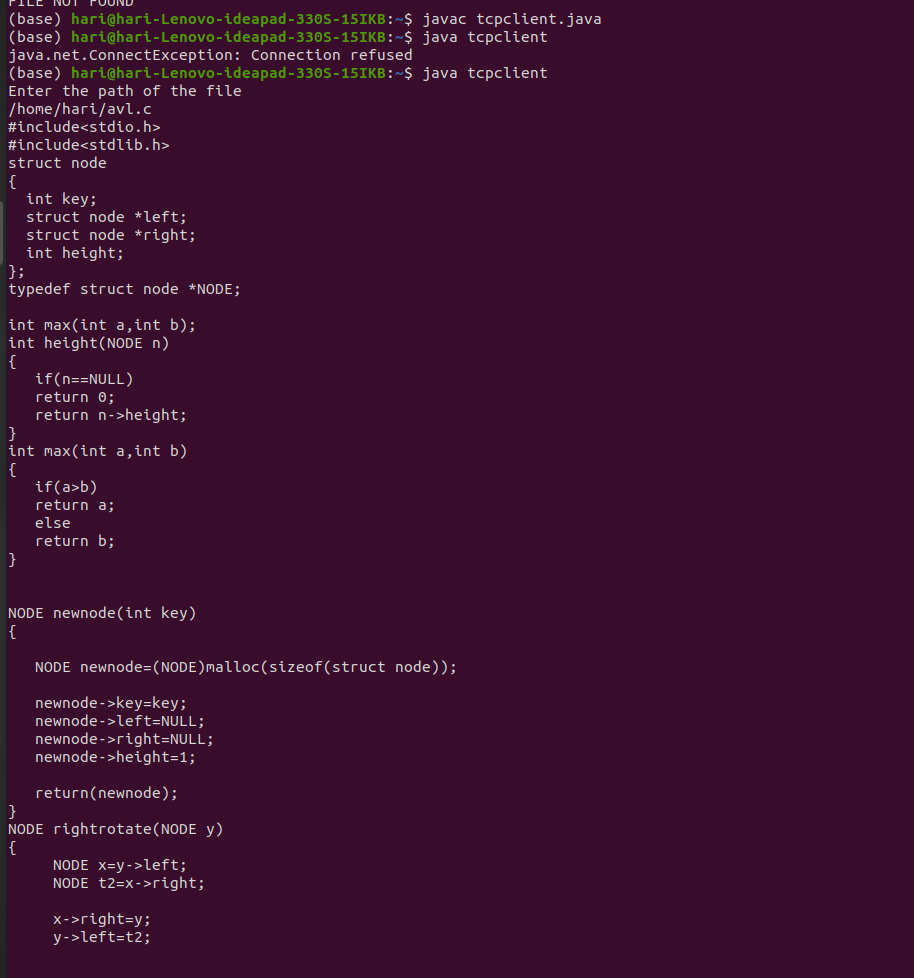
System.out.println(e);

}

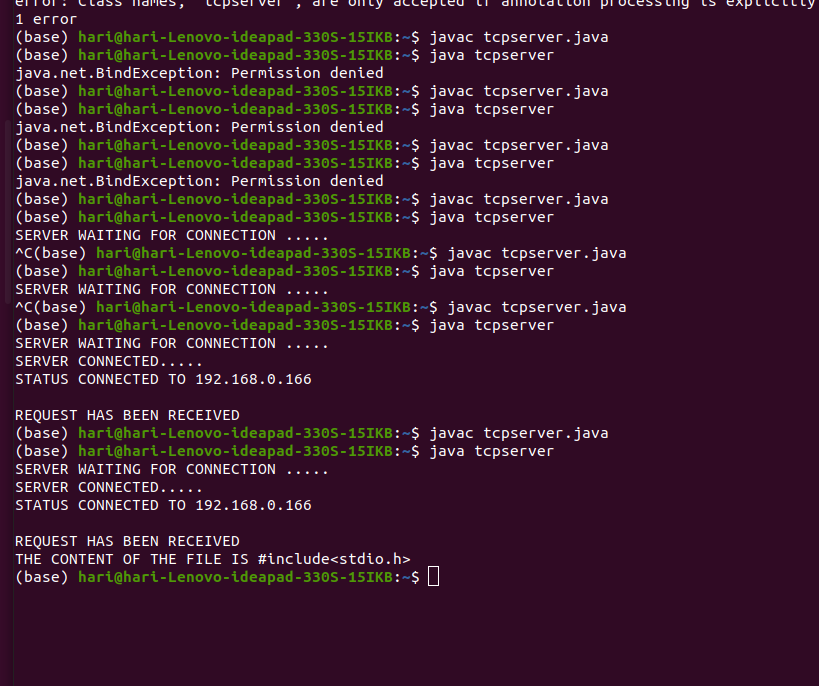
}

}

TCP CLIENT



TCP SERVER



6:Using UDP sockets, write a client-server program to make client sending the

file name and the server to send back the contents of the requested file if

present.

import java.util.\*;

import java.io.\*;

import java.lang.\*;

import java.net.\*;

public class udpclient

{

public static void main(String[] args)

{

try

{

InetAddress address=InetAddress.getByName("127.0.0.1");

int port=2000;

DatagramSocket socket=new DatagramSocket();

Scanner sc=new Scanner(System.in);

String name;

System.out.println("Enter the name of the file");

name=sc.nextLine();

byte[] buffer=name.getBytes();

DatagramPacket request=new DatagramPacket(buffer,buffer.length,address,port);

socket.send(request);

byte[] respo=new byte[512];

DatagramPacket response=new DatagramPacket(respo,respo.length);

socket.receive(response);

String cont=new String(respo,0,response.getLength());

System.out.println(cont);

}

catch(Exception e)

{

System.out.println(e);

}

}

}

import java.io.\*;

import java.lang.\*;

import java.net.\*;

import java.util.\*;

public class udpserver

{

public static void main(String[] args)

{

try{

byte[] buffer=new byte[512];

DatagramSocket socket=new DatagramSocket(2000);

DatagramPacket request = new DatagramPacket(buffer, buffer.length);

socket.receive(request);

String path = new String(buffer, 0, request.getLength());

System.out.println("REQUEST RECEIVED");

try

{

File mf=new File(path);

Scanner sc=new Scanner(mf);

String nt=sc.nextLine();

while(sc.hasNextLine())

{

nt=nt+"\n" + sc.nextLine();

}

//System.out.println(nt);

InetAddress clientAddress=request.getAddress();

int clientPort=request.getPort();

byte[] ans=nt.getBytes();

DatagramPacket response = new DatagramPacket(ans, ans.length, clientAddress,clientPort);

socket.send(response);

}

catch(FileNotFoundException e)

{

String resp="FILE NOT FOUND";

InetAddress clientAddress=request.getAddress();

int clientPort=request.getPort();

byte[] ans=resp.getBytes();

DatagramPacket response = new DatagramPacket(ans, ans.length, clientAddress,clientPort);

socket.send(response);

}

}catch(Exception e)

{

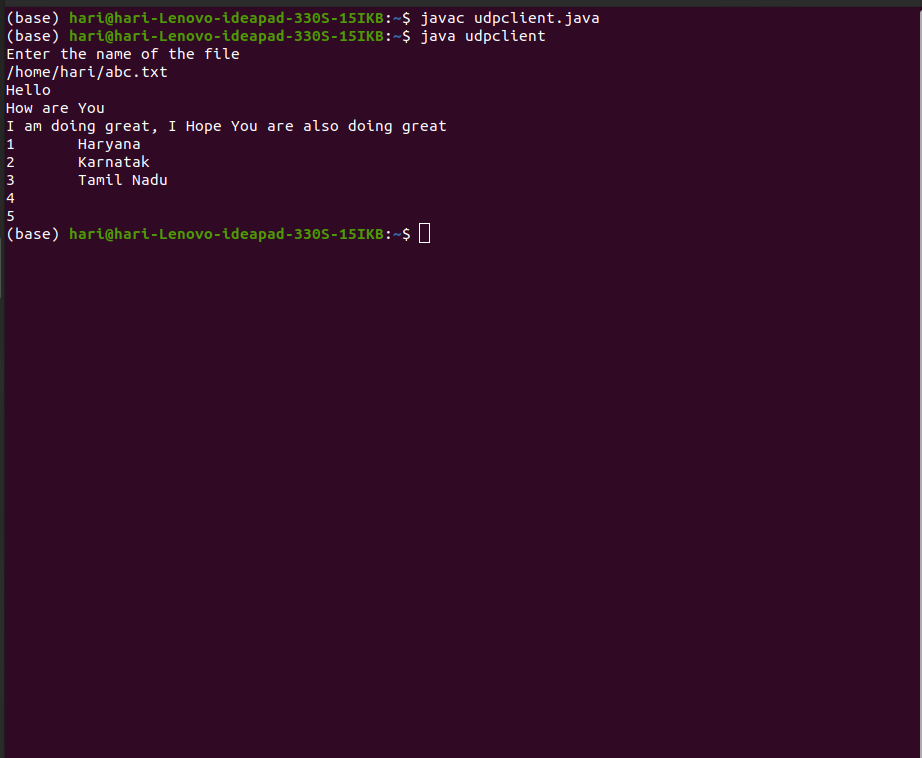
System.out.println(e);

}

}

}

UDP CLIENT:



UDP SERVER:

