

INDEX

EXP NO	TITLE	PAGE No	DATE	SIG
1	BASIC NETWORKING COMMANDS	2		
2	CLIENT SERVER COMMUNICATION (SOCKET PROGRAMMING)	7		
3	CODE FOR SIMULATING PING AND TRACEROUTE COMMANDS	10		
4	CREATE SOCKET FOR HTTP, TO WEBPAGE UPLOAD AND DOWNLOAD	12		
5	SIMULATION OF ERROR CORRECTION CODE (CRC)	15		
6	IMPLEMENTATION OF STOP AND WAIT PROTOCOL	17		
7	SIMULATION OF SLIDING WINDOW PROTOCOL	21		
8	SIMULATING ARP PROTOCOLS	26		
9	PROGRAM FOR REVERSE ADDRESS RESOLUTION PROTOCOL USING UDP	29		
10	ECHO CLIENT AND ECHO SERVER	33		
11	CHAT	37		
12	FILE TRANSFER	40		
13	SIMULATION OF DNS USING UDP SOCKETS	43		
14	B. SNMP APPLICATION	47		

EXPERIMENT-1

BASIC NETWORKING COMMANDS

AIM: To implement basic networking keywords

PROCEDURE:

1. Study the basic networking commands.
2. Run the networking commands along with their arguments and parameters.
3. Observe the output on the command line.
4. Record observations in the journal.

SOURCE CODE:

1. Tracert :

The **tracert** command prints the path. If all routers on the path are functional, this command prints the full path. If a router is down on the path, this command prints the path up to the last operational router.

tracert **www.google.co.in**

2. Ping :

The **ping** command is used to test connectivity between two hosts.

ping **google.com**

3. Arp :

By default, this command displays the ARP table of the active NIC. If multiple NICs are installed on the computer, you can use the **-a** option with this command. If the **-a** option is used, the ARP command displays all ARP tables .

arp -a

4. netstat :

This command displays active connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, and IP statistics.

Netstat

5. Ipconfig :

This command displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings. This command is mainly used to view the IP addresses on the computers that are configured to obtain their IP address automatically.

ipconfig

OUTPUT:

```
Command Prompt
Microsoft Windows [Version 10.0.22000.376]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>tracert www.google.com

Tracing route to www.google.com [142.250.195.164]
over a maximum of 30 hops:

  0  3 ms    4 ms    2 ms  dlinkrouter [192.168.0.1]
  1  5 ms    3 ms    4 ms  172.16.134.1
  2  14 ms   15 ms   13 ms  103.143.169.81
  3  15 ms   16 ms   14 ms  nsg-static-005.166.72.182.airtel.in [182.72.166.5]
  4  *        27 ms   *      116.119.57.100
  5  39 ms   41 ms   40 ms  72.14.216.192
  6  35 ms   35 ms   36 ms  216.239.43.131
  7  49 ms   41 ms   41 ms  142.251.55.89
  8  39 ms   38 ms   38 ms  maa03s41-in-f4.1e100.net [142.250.195.164]

Trace complete.

C:\Users\DELL>ping www.google.com

Pinging www.google.com [142.250.195.164] with 32 bytes of data:
Reply from 142.250.195.164: bytes=32 time=39ms TTL=119
Reply from 142.250.195.164: bytes=32 time=39ms TTL=119
Reply from 142.250.195.164: bytes=32 time=39ms TTL=119
Reply from 142.250.195.164: bytes=32 time=38ms TTL=119

Ping statistics for 142.250.195.164:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 38ms, Maximum = 39ms, Average = 38ms

C:\Users\DELL>arp -a

Interface: 192.168.0.109 --- 0xf
Internet Address      Physical Address      Type
192.168.0.1           bc-0f-9a-ed-12-1c    dynamic
192.168.0.255         ff-ff-ff-ff-ff-ff    static
224.0.0.2             01-00-5e-00-00-02    static
224.0.0.22            01-00-5e-00-00-16    static
224.0.0.251           01-00-5e-00-00-fb    static
224.0.0.252           01-00-5e-00-00-fc    static
228.8.8.8             01-00-5e-08-08-08    static
239.255.255.250       01-00-5e-7f-ff-fa    static
255.255.255.255       ff-ff-ff-ff-ff-ff    static

C:\Users\DELL>netstat
```

Command Prompt

Active Connections

Proto	Local Address	Foreign Address	State
TCP	127.0.0.1:49671	DESKTOP-AU0PH1F:49672	ESTABLISHED
TCP	127.0.0.1:49672	DESKTOP-AU0PH1F:49671	ESTABLISHED
TCP	127.0.0.1:49673	DESKTOP-AU0PH1F:49674	ESTABLISHED
TCP	127.0.0.1:49674	DESKTOP-AU0PH1F:49673	ESTABLISHED
TCP	127.0.0.1:49678	DESKTOP-AU0PH1F:49679	ESTABLISHED
TCP	127.0.0.1:49679	DESKTOP-AU0PH1F:49678	ESTABLISHED
TCP	127.0.0.1:49680	DESKTOP-AU0PH1F:49681	ESTABLISHED
TCP	127.0.0.1:49681	DESKTOP-AU0PH1F:49680	ESTABLISHED
TCP	127.0.0.1:49682	DESKTOP-AU0PH1F:49683	ESTABLISHED
TCP	127.0.0.1:49683	DESKTOP-AU0PH1F:49682	ESTABLISHED
TCP	127.0.0.1:49684	DESKTOP-AU0PH1F:49685	ESTABLISHED
TCP	127.0.0.1:49685	DESKTOP-AU0PH1F:49684	ESTABLISHED
TCP	192.168.0.109:49286	bom12s11-in-f14:https	ESTABLISHED
TCP	192.168.0.109:49295	bom07s27-in-f14:https	ESTABLISHED
TCP	192.168.0.109:49362	whatsapp-cdn-shv-02-bom1:https	ESTABLISHED
TCP	192.168.0.109:49486	bom12s15-in-f1:https	ESTABLISHED
TCP	192.168.0.109:49494	maa03s44-in-f2:https	ESTABLISHED
TCP	192.168.0.109:49498	server-13-227-178-25:https	ESTABLISHED
TCP	192.168.0.109:49501	bom07s18-in-f3:https	ESTABLISHED
TCP	192.168.0.109:49502	maa03s41-in-f4:https	CLOSE_WAIT
TCP	192.168.0.109:49503	maa03s41-in-f4:https	CLOSE_WAIT
TCP	192.168.0.109:49504	20.42.65.90:https	ESTABLISHED
TCP	192.168.0.109:49505	20.42.65.90:https	ESTABLISHED
TCP	192.168.0.109:49506	20.42.65.90:https	ESTABLISHED
TCP	192.168.0.109:49507	a-0003:https	ESTABLISHED
TCP	192.168.0.109:65493	20.197.71.89:https	ESTABLISHED
TCP	192.168.0.109:65518	sa-in-f188:5228	ESTABLISHED
TCP	192.168.0.109:65519	maa05s13-in-f10:https	ESTABLISHED
TCP	192.168.0.109:65522	maa05s13-in-f10:https	ESTABLISHED

C:\Users\DELL>ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

Media State : Media disconnected
Connection-specific DNS Suffix . :

Unknown adapter Local Area Connection:

Media State : Media disconnected
Connection-specific DNS Suffix . :

Command Prompt

Wireless LAN adapter Local Area Connection* 1:

Media State : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 10:

Media State : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :
IPv6 Address : fd01::a8c5:c714:e05a:ba2
Temporary IPv6 Address : fd01::c818:b23e:45b1:ef7b
Link-local IPv6 Address : fe80::a8c5:c714:e05a:ba2%15
IPv4 Address : 192.168.0.109
Subnet Mask : 255.255.255.0
Default Gateway : fe80::be0f:9aff:feed:121c%15
192.168.0.1

C:\Users\DELL>tracert

Usage: tracert [-d] [-h maximum_hops] [-j host-list] [-w timeout]
[-R] [-S srcaddr] [-4] [-6] target_name

Options:

-d Do not resolve addresses to hostnames.
-h maximum_hops Maximum number of hops to search for target.
-j host-list Loose source route along host-list (IPv4-only).
-w timeout Wait timeout milliseconds for each reply.
-R Trace round-trip path (IPv6-only).
-S srcaddr Source address to use (IPv6-only).
-4 Force using IPv4.
-6 Force using IPv6.

C:\Users\DELL>tracert[-d]

'tracert[-d]' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\DELL>tracert [-d]

Unable to resolve target system name [-d].

C:\Users\DELL>tracert [-h]

Unable to resolve target system name [-h].

C:\Users\DELL>ipconfig/all

Command Prompt

C:\Users\DELL>ipconfig/all

Windows IP Configuration

Host Name : DESKTOP-AU0PH1F
Primary Dns Suffix :
Node Type : Hybrid
IP Routing Enabled. : No
WINS Proxy Enabled. : No

Ethernet adapter Ethernet:

Media State : Media disconnected
Connection-specific DNS Suffix . :
Description : Realtek PCIe FE Family Controller
Physical Address. : 54-48-10-ED-99-B2
DHCP Enabled. : Yes
Autoconfiguration Enabled : Yes

Unknown adapter Local Area Connection:

Media State : Media disconnected
Connection-specific DNS Suffix . :
Description : TAP-Windows Adapter V9
Physical Address. : 00-FF-93-50-DA-00
DHCP Enabled. : Yes
Autoconfiguration Enabled : Yes

Wireless LAN adapter Local Area Connection* 1:

Media State : Media disconnected
Connection-specific DNS Suffix . :
Description : Microsoft Wi-Fi Direct Virtual Adapter
Physical Address. : 82-2B-F9-5C-C3-51
DHCP Enabled. : Yes
Autoconfiguration Enabled : Yes

Wireless LAN adapter Local Area Connection* 10:

Media State : Media disconnected
Connection-specific DNS Suffix . :
Description : Microsoft Wi-Fi Direct Virtual Adapter #2
Physical Address. : 92-2B-F9-5C-C3-51
DHCP Enabled. : No
Autoconfiguration Enabled : Yes

Wireless LAN adapter Wi-Fi:

Command Prompt

Connection-specific DNS Suffix . :
Description : Qualcomm QCA9377 802.11ac Wireless Adapter
Physical Address. : 80-2B-F9-5C-C3-51
DHCP Enabled. : Yes
Autoconfiguration Enabled : Yes
IPv6 Address : fd01::a8c5:c714:e05a:ba2(Preferred)
Temporary IPv6 Address. : fd01::c818:b23e:45b1:ef7b(Preferred)
Link-local IPv6 Address : fe80::a8c5:c714:e05a:ba2%15(Preferred)
IPv4 Address. : 192.168.0.109(Preferred)
Subnet Mask : 255.255.255.0
Lease Obtained. : 09 January 2022 22:44:56
Lease Expires : 11 January 2022 07:50:44
Default Gateway : fe80::be0f:9aff:feed:121c%15
192.168.0.1
DHCP Server : 192.168.0.1
DHCPv6 IAID : 142617593
DHCPv6 Client DUID. : 00-01-00-01-29-15-7F-FD-54-48-10-ED-99-B2
DNS Servers : fe80::be0f:9aff:feed:121c%15
192.168.0.1
NetBIOS over Tcpi. : Enabled

C:\Users\DELL>ipconfig/renew

Windows IP Configuration

No operation can be performed on Ethernet while it has its media disconnected.
No operation can be performed on Local Area Connection while it has its media disconnected.
No operation can be performed on Local Area Connection* 1 while it has its media disconnected.
No operation can be performed on Local Area Connection* 10 while it has its media disconnected.

Ethernet adapter Ethernet:

Media State : Media disconnected
Connection-specific DNS Suffix . :

Unknown adapter Local Area Connection:

Media State : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 1:

Media State : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 10:

Media State : Media disconnected

```
Command Prompt
Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :
IPv6 Address. . . . . : fd01::a8c5:c714:e05a:ba2
Temporary IPv6 Address. . . . . : fd01::c818:b23e:45b1:ef7b
Link-local IPv6 Address . . . . . : fe80::a8c5:c714:e05a:ba2%15
IPv4 Address. . . . . : 192.168.0.109
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::be0f:9aff:feed:121c%15
                          192.168.0.1

C:\Users\DELL>ipconfig/release

Windows IP Configuration

No operation can be performed on Ethernet while it has its media disconnected.
No operation can be performed on Local Area Connection while it has its media disconnected.
No operation can be performed on Local Area Connection* 1 while it has its media disconnected.

Ethernet adapter Ethernet:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Unknown adapter Local Area Connection:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 1:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Local Area Connection* 10:

Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . :

Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . :
IPv6 Address. . . . . : fd01::a8c5:c714:e05a:ba2
Temporary IPv6 Address. . . . . : fd01::c818:b23e:45b1:ef7b
Link-local IPv6 Address . . . . . : fe80::a8c5:c714:e05a:ba2%15
Default Gateway . . . . . : fe80::be0f:9aff:feed:121c%15
```

RESULT: Basic Networking Commands are implemented successfully.

EXPERIMENT-2

CLIENT SERVER COMMUNICATION (SOCKET PROGRAMMING)

AIM: To Implement Client Server Communication

PROCEDURE:

CLIENT SIDE

1. Start the program.
2. Create a socket which binds the Ip address of server and the port address to acquire service.
3. After establishing connection send a data to server.
4. Receive and print the same data from server.
5. Close the socket.
6. End the program.

SERVER SIDE

1. Start the program.
2. Create a server socket to activate the port address.
3. Create a socket for the server socket which accepts the connection.
4. After establishing connection receive the data from client.
5. Print and send the same data to client.
6. Close the socket.
7. End the program.

SOURCE CODE:

Server

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

class dateserver
{
    public static void main(String args[])
    {
        ServerSocket ss;
        Socket s;
        PrintStream ps;
```

```

        DataInputStream dis;
        String inet;
        try
        {
            ss = new ServerSocket(8020);
            while (true)
            {
                s = ss.accept();
                ps = new PrintStream(s.getOutputStream());
                Date d = new Date();
                ps.println(d);
                dis = new DataInputStream(s.getInputStream());
                inet = dis.readLine();
                System.out.println("THE CLIENT SYSTEM ADDRESS IS:" + inet);
                ps.close();
            }
        }
        catch (IOException e)
        {
            System.out.println("The Exception is :" + e);
        }
    }
}

```

Client

```

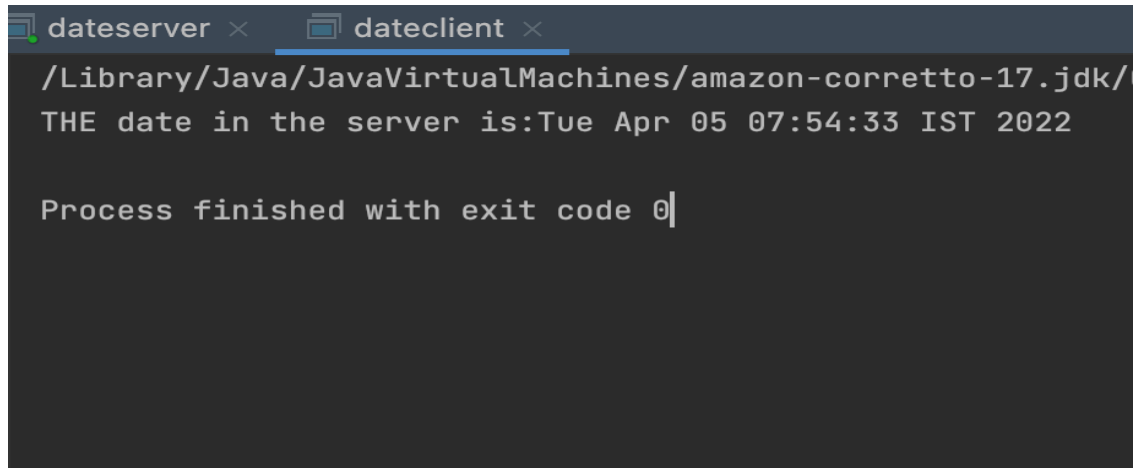
import java.net.*;
import java.io.*;
class dateclient
{
    public static void main(String[] args)
    {
        Socket soc;
        DataInputStream dis;
        String sdate;
        PrintStream ps;
        try
        {
            InetAddress ia=InetAddress.getLocalHost();
            soc=new Socket(ia,8020);
            dis=new DataInputStream(soc.getInputStream());
            sdate=dis.readLine();
            System.out.println("THE date in the server is:"+sdate);
            ps=new PrintStream(soc.getOutputStream());
            ps.println(ia);
        }
        catch(IOException e)
        {
            System.out.println("THE EXCEPTION is: "+e);
        }
    }
}

```

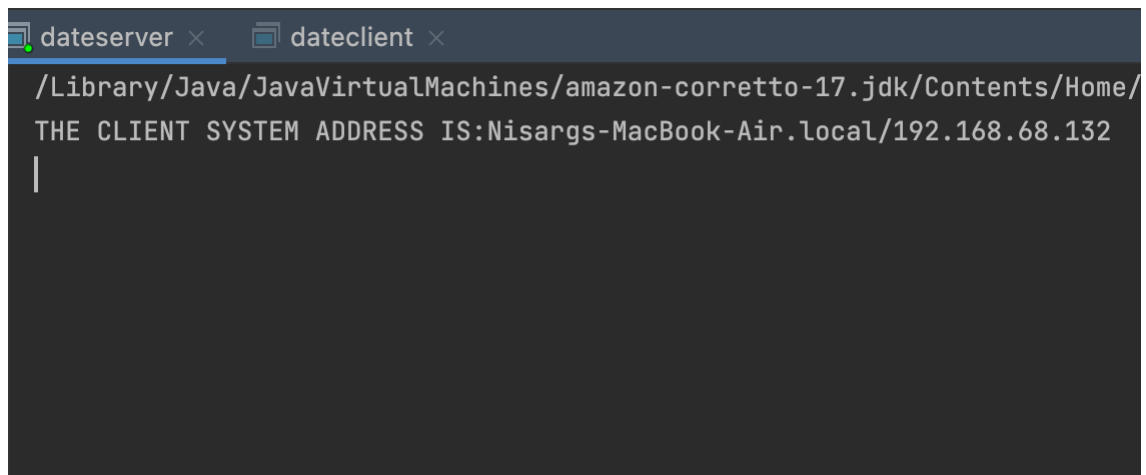


```
}  
}
```

OUTPUT:



```
dateserver x dateclient x  
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/  
THE date in the server is:Tue Apr 05 07:54:33 IST 2022  
  
Process finished with exit code 0|
```



```
dateserver x dateclient x  
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/Contents/Home/  
THE CLIENT SYSTEM ADDRESS IS:Nisargs-MacBook-Air.local/192.168.68.132  
|
```

RESULT: Client Server Communication using socket programming has been implemented successfully.

EXPERIMENT-3

CODE FOR SIMULATING PING AND TRACEROUTE COMMANDS

AIM: To Write The java program for simulating ping and traceroute commands

PROCEDURE:

- 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

SOURCE CODE:

Server

```
import java.io.*;
import java.net.*;
class pingserver
{
public static void main(String args[])
{
try
{
String str;
System.out.print(" Enter the IP Address to be Ping : ");
DataInputStream dis=new DataInputStream(System.in);
String ip=dis.readLine();
Runtime H=Runtime.getRuntime();
Process p=H.exec("ping " + ip);
InputStream in=p.getInputStream();
DataInputStream buf2=new DataInputStream(in);
while((str=buf2.readLine())!=null)
{
System.out.println(" " + str);
}
}
}
catch(Exception e)
{

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

Client

```

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

public class traceroutecmd
{
    public static void runSystemCommand(String command)
    {
        try
        {
            Process p = Runtime.getRuntime().exec(command);
            DataInputStream inputstream=new DataInputStream(p.getInputStream());

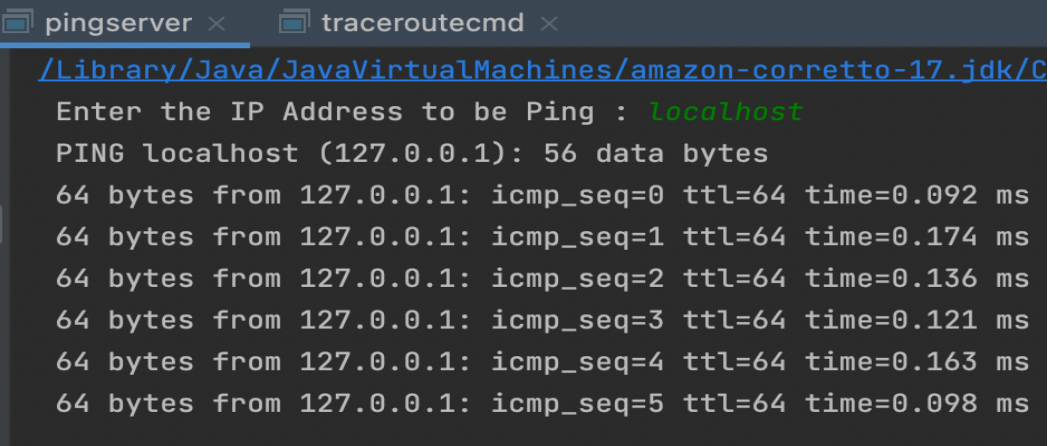
            String s = "";
            while ((s = inputstream.readLine()) != null)
                System.out.println(s);
        }
        catch (Exception e)
        {
        }
    }

    public static void main(String[] args)
    {
        String ip = "www.google.co.in";

        runSystemCommand("tracert " + ip);
    }
}

```

OUTPUT



```

/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/C
Enter the IP Address to be Ping : localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.092 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.174 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.136 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.121 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.163 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.098 ms

```

RESULT:

Thus the program was implementing to simulating ping and traceroute commands.

EXPERIMENT-4

CREATE SOCKET FOR HTTP,TO WEBPAGE UPLOAD AND DOWNLOAD

AIM: To write a java program for socket for HTTP for web page upload and download .

PROCEDURE:

- 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

SOURCE CODE:

Server

```
import java.net.*;
import java.io.*;
import java.awt.image.*;
import javax.imageio.*;
import javax.swing.*;

class Server {
    public static void main(String args[]) throws Exception{
        ServerSocket server=null;
        Socket socket;
        server=new ServerSocket(4000);
        System.out.println("Server Waiting for image");
        socket=server.accept(); System.out.println("Client connected.");
        InputStream in = socket.getInputStream();
        DataInputStream dis = new DataInputStream(in);
        int len = dis.readInt();
        System.out.println("Image Size: " + len/1024 + "KB"); byte[] data = new byte[len];
        dis.readFully(data);
        dis.close();
        in.close();
        InputStream ian = new ByteArrayInputStream(data);
        BufferedImage bImage = ImageIO.read(ian);
        JFrame f = new JFrame("Server");
        ImageIcon icon = new ImageIcon(bImage);
        JLabel l = new JLabel();
        l.setIcon(icon);
        f.add(l);
        f.pack();
        f.setVisible(true);
    }
}
```

```
}}
```

Client

```
import javax.swing.*;
import java.net.*;
import java.awt.image.*;
import javax.imageio.*;
import java.io.*;
import java.awt.image.BufferedImage;
import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException; import
javax.imageio.ImageIO;
public class Client{
    public static void main(String args[]) throws Exception{
        Socket soc;
        BufferedImage img = null;
        soc=new Socket("localhost",4000);
        System.out.println("Client is running. ");
        try {
            System.out.println("Reading image from disk. ");
            img = ImageIO.read(new File("dog.jpg"));
            ByteArrayOutputStream baos = new ByteArrayOutputStream();
            ImageIO.write(img, "jpg", baos);
            baos.flush();
            byte[] bytes = baos.toByteArray();
            baos.close();
            System.out.println("Sending image to server. ");
            OutputStream out = soc.getOutputStream();
            DataOutputStream dos = new DataOutputStream(out);
            dos.writeInt(bytes.length);
            dos.write(bytes, 0, bytes.length);
            System.out.println("Image sent to server. ");
            dos.close();
            out.close();
        } catch (Exception e) { System.out.println("Exception: " + e.getMessage());
            soc.close();
        }
        soc.close();
    }
}
```

OUTPUT

```
Server x Client x
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/Contents
Client is running.
Reading image from disk.
Sending image to server.
Image sent to server.

Process finished with exit code 0
|
```

```
Server x Client x
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/Contents
Server Waiting for image
Client connected.
Image Size: 426KB

Process finished with exit code 0
```

Server			
		Security	Sec
9:45 AM	10:35 AM	Enterprise Systems	Indian /
10:40 AM	11:30 AM	Enterprise Systems	Indian /
11:30 AM	12:30 PM		Art Intel
12:30 PM	1:20 PM	Artificial Intelligence	
1:25 PM	2:15 PM	Artificial Intelligence	
2:20 PM	3:10 PM		

RESULT: Thus the program was implementing to socket for HTTP for web page upload and download.

EXPERIMENT-5

SIMULATION OF ERROR CORRECTION CODE (CRC)

AIM: To Simulation of Error Correction Code using C++

PROCEDURE:

1. Open the editor and type the program for error detection
2. Get the input in the form of bits.
3. Append the redundancy bits.
4. Divide the appended data using a divisor polynomial.
5. The resulting data should be transmitted to the receiver.
6. At the receiver the received data is entered.
7. The same process is repeated at the receiver.
8. If the remainder is zero there is no error otherwise there is some error in the received bits
9. Run the program.

SOURCE CODE:

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

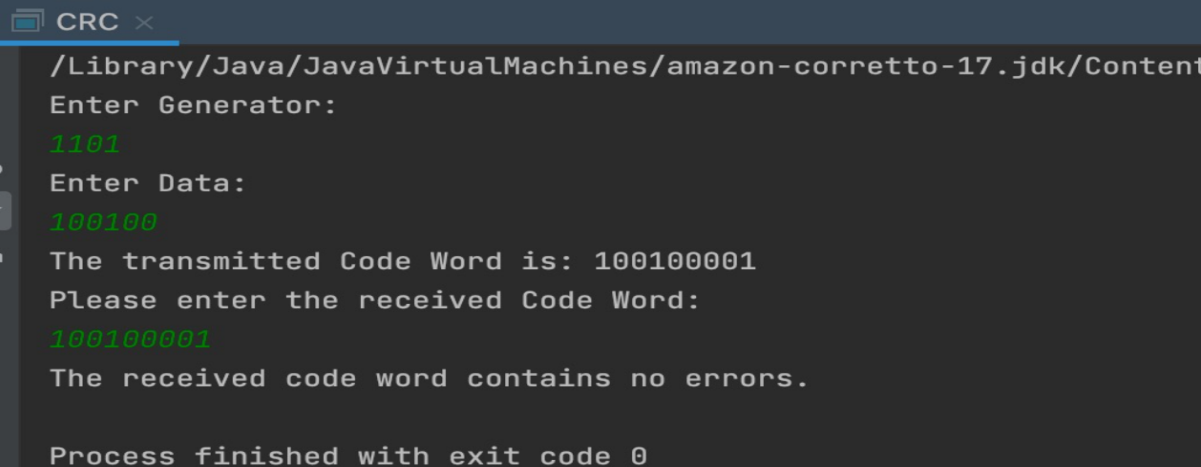
System.out.println("Enter Generator:");
DataInputStream br=new DataInputStream(System.in);
String gen = br.readLine();
System.out.println("Enter Data:");
String data = br.readLine();
String code = data;
while(code.length() < (data.length() + gen.length() - 1))
{
code = code + "0";
}
code = data + div(code,gen);
System.out.println("The transmitted Code Word is: " + code);
System.out.println("Please enter the received Code Word: ");
String rec = br.readLine();
if(Integer.parseInt(div(rec,gen)) == 0)
System.out.println("The received code word contains no errors.");
else
System.out.println("The received code word contains errors.");
}
static String div(String num1,String num2)
```

```

{
int pointer = num2.length();
String result = num1.substring(0, pointer);
String remainder = "";
for(int i = 0; i < num2.length(); i++)
{
if(result.charAt(i) == num2.charAt(i))
remainder += "0";
else
remainder += "1";
}
while(pointer < num1.length())
{
if(remainder.charAt(0) == '0')
{
remainder = remainder.substring(1, remainder.length());
remainder = remainder + String.valueOf(num1.charAt(pointer));
pointer++;
}
result = remainder;
remainder = "";
for(int i = 0; i < num2.length(); i++)
{
if(result.charAt(i) == num2.charAt(i))
remainder += "0";
else
remainder += "1";
}
}
return remainder.substring(1,remainder.length());
}
}

```

OUTPUT:



```

CRC x
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/Content
Enter Generator:
1101
Enter Data:
100100
The transmitted Code Word is: 100100001
Please enter the received Code Word:
100100001
The received code word contains no errors.

Process finished with exit code 0

```


RESULT: Thus the error detection and error correction is implemented successfully.

EXPERIMENT-6

IMPLEMENTATION OF STOP AND WAIT PROTOCOL

AIM: To write a java program to perform stop and wait protocol

PROCEDURE:

- 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

SOURCE CODE:

Server

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

import java.util.*;

public class receiver {

    public static void main(String args[])

    {

        String h="Serverhost";

        int q=5000;

        int i;

        try

        {

            ServerSocket ss2;

            ss2 = new ServerSocket(8000);

            Socket s1 =ss2.accept();
```

```

        DataInputStream dd1= new DataInputStream(s1.getInputStream());

        Integer i1 =dd1.read();

        for(i=0;i<i1;i++)

        {

            ServerSocket ss1;

            ss1 = new ServerSocket(9000+i);

            Socket s =ss1.accept();

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

            String sss1 = dd.readUTF();

            System.out.println(sss1);

            System.out.println("Frame "+ i+" received");

            DataOutputStream d1 = new DataOutputStream(s.getOutputStream());

            d1.write(i);

            System.out.println("ACK sent for "+ i);

        }

    }

    catch(Exception ex)

    {

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

    }

}

```

Client

```

import java.io.*;
import java.net.*;
import java.util.Scanner;
public class sender {

```

```

public static void main(String args[])
{
    int p=9000,i,q=8000;
    String h="localhost";
    try
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter number of frames : ");
        int number = scanner.nextInt();
        if(number==0)
        {
            System.out.println("No frame is sent");
        }
        else
        {
            Socket s2= new Socket(h,q);

            DataOutputStream d1 = new DataOutputStream(s2.getOutputStream());

            d1.write(number);

        }

        String str1;

        for (i=0;i<number;i++)
        {

            System.out.print("Enter message : ");

            String name = scanner.next();

            System.out.println("Frame " + i+" is sent");

            Socket s1;

            s1= new Socket(h,p+i);

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

            d.writeUTF(name);

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

            Integer sss1 = dd.read();

            System.out.println("Ack for :"+ sss1 + " is received");

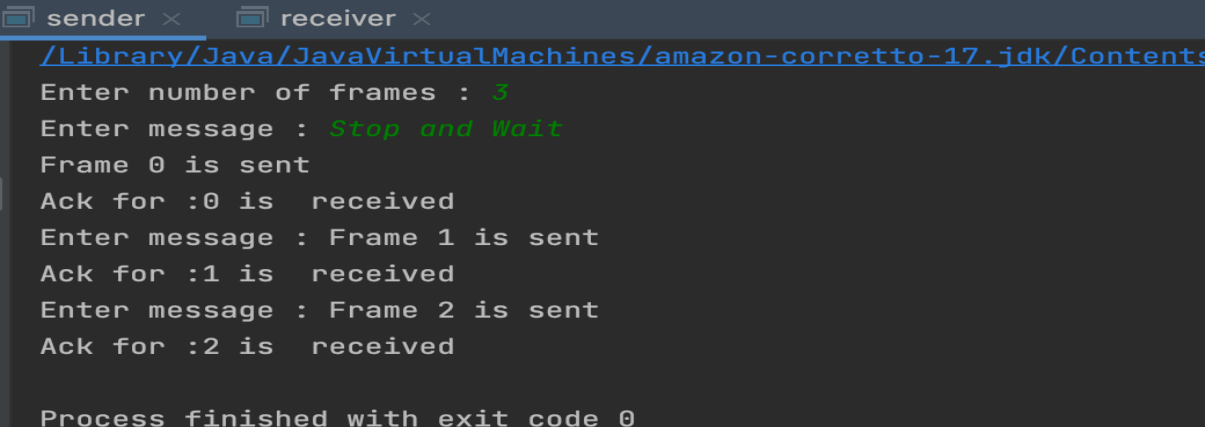
```

```

    }
}
catch(Exception ex)
{
    System.out.println("ERROR :"+ex);
}
}
}

```

OUTPUT:

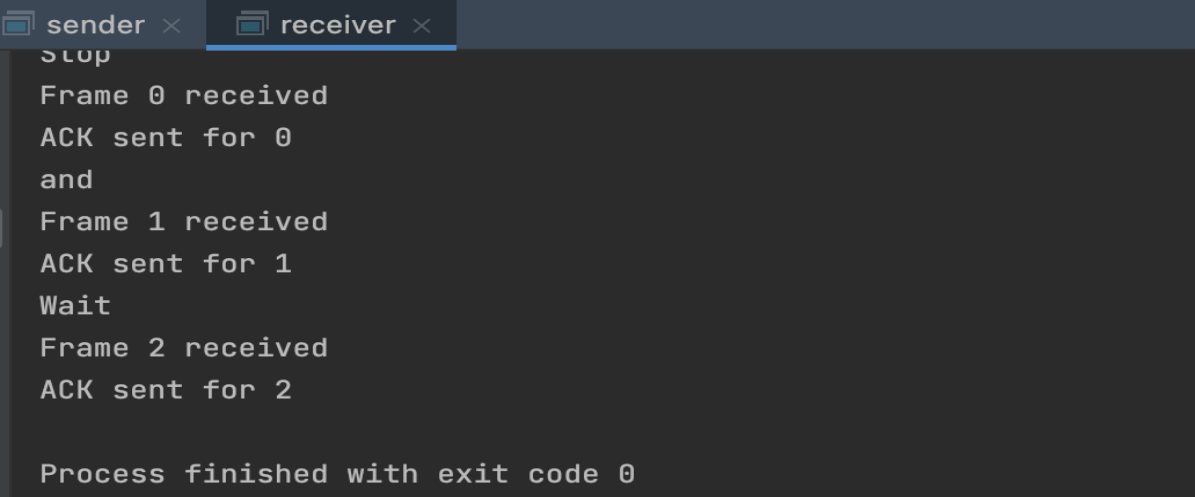


```

sender x receiver x
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/Contents
Enter number of frames : 3
Enter message : Stop and Wait
Frame 0 is sent
Ack for :0 is received
Enter message : Frame 1 is sent
Ack for :1 is received
Enter message : Frame 2 is sent
Ack for :2 is received

Process finished with exit code 0

```



```

sender x receiver x
Stop
Frame 0 received
ACK sent for 0
and
Frame 1 received
ACK sent for 1
Wait
Frame 2 received
ACK sent for 2

Process finished with exit code 0

```

RESULT: Thus the program for implementing stop and wait protocol was executed successfully.

EXPERIMENT-7

SIMULATION OF SLIDING WINDOW PROTOCOL

AIM: To Implement Simulation of Sliding Window Protocol

PROCEDURE:

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 the server from the client-side.
5. If your frames reach the server it will send ACK signal to client otherwise it will send a NACK signal to the client.
6. Stop the program

SOURCE CODE:

Server

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

public class slidesserver
{
    public static void main(String args[])throws IOException
    {
        byte msg[]=new byte[200];
        ServerSocket ser=new ServerSocket(2000);
        Socket s;
        PrintStream pout;
```

```

DataInputStream in=new DataInputStream(System.in);

int start,end,l,j=0,i,ws=5,k=0;

String st,stl[]=new String[100];

System.out.println("Type\"Stop\"to exit");

s=ser.accept();

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

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

System.out.println("enter data to be send:");

while(true)

{

st=in.readLine();

l=st.length();

start=0;

end=10;

j=0;

if(st.equals("STOP"))

{

pout.println(st);

break;

}

if(l<10)

{stl[j++]=l+st;

}

else

{

for(i=l,j=0;i>0;i=i-10,j++)

{

```

```

    stl[j]=(j+1)+st.substring(start,end);
    start=end;
    end=end+10;
    if(end>l)
    {
        end=((start-10)+i);
    }
    System.out.println("total number of packet"+j);
}
pout.println(j);
for(i=0;i<j;i++)
{
    pout.println(stl[i]);
    if((i+1)%ws==0)
    {
        System.out.println(rin.readLine());
    }
    if(i%ws!=0)
    {
        System.out.println(rin.readLine());
        System.out.println("enter next data to send");
    }
}
}
}
}

```

Client

```

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

public class slideclient

```

```
{
```

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

```
{
```

```
    byte msg[]=new byte[200];
```

```
    Socket s=new Socket(InetAddress.getLocalHost(),2000);
```

```
    DataInputStream in=new DataInputStream(s.getInputStream());
```

```
    DataInputStream ain=new DataInputStream(System.in);
```

```
    PrintStream p=new PrintStream(s.getOutputStream());
```

```
    char ch;
```

```
    int i=0,ws=5;
```

```
    while(true)
```

```
    {
```

```
        String str,astr;
```

```
        int j=0;
```

```
        i=0;
```

```
        str=in.readLine();
```

```
        if(str.equals("STOP"))
```

```
            System.exit(0);
```

```
        j=Integer.parseInt(str);
```

```
        for(i=0;i<j;i++)
```

```
        {
```

```
            str=in.readLine();
```

```
            System.out.println(str);
```

```
            if((i+1)%ws==0)
```

```
            {
```

```
                System.out.println("Give the ack by press\"Enter\"key");
```

```
                astr=ain.readLine();
```



```

        p.println((i+1)+"ack");

    }}

    if((i%ws)!=0)

    {

        System.out.println("Give the ack by press\"Enter\"key");

        astr=ain.readLine();

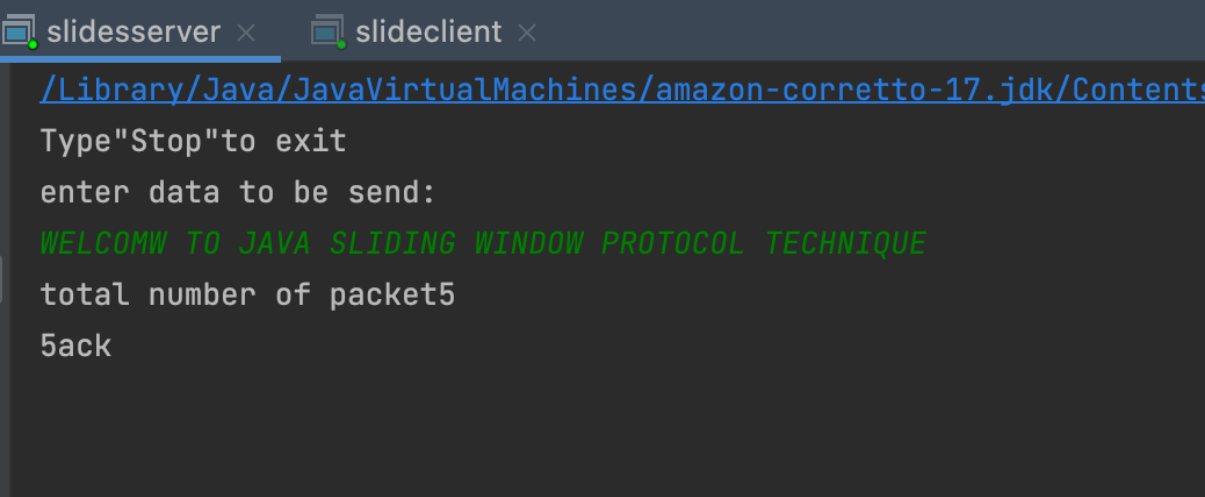
        p.println(i+"ack");}

    System.out.println("All data are recieved and ack");

}}}

```

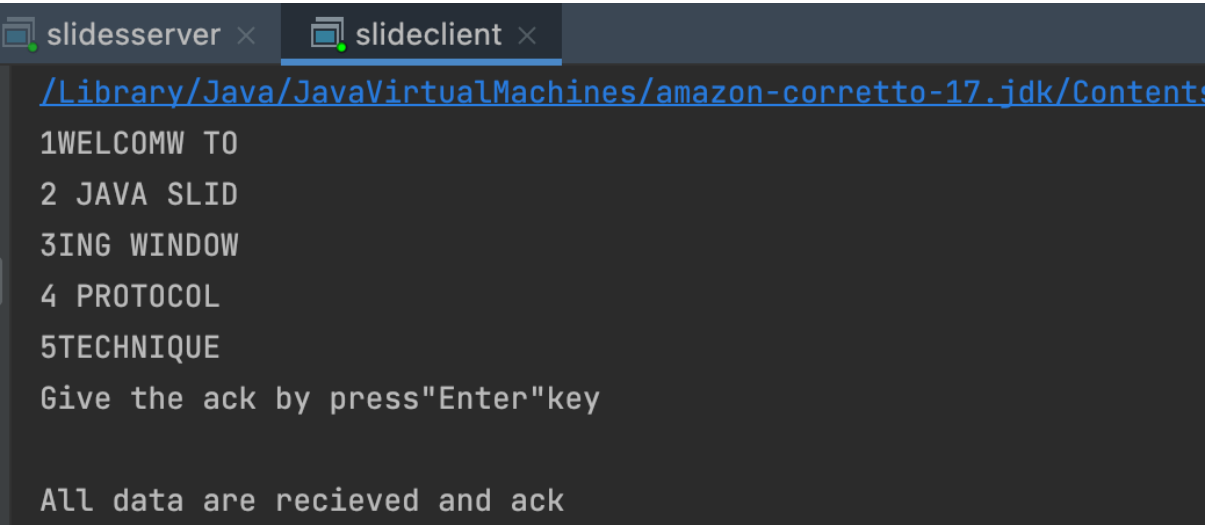
OUTPUT:



```

slidesserver x slideclient x
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/Content.txt
Type"Stop"to exit
enter data to be send:
WELCOMW TO JAVA SLIDING WINDOW PROTOCOL TECHNIQUE
total number of packet5
5ack

```



```

slidesserver x slideclient x
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/Content.txt
1WELCOMW TO
2 JAVA SLID
3ING WINDOW
4 PROTOCOL
5TECHNIQUE
Give the ack by press"Enter"key

All data are recieved and ack

```

RESULT: Thus, the above program sliding window protocol was executed and successfully.

EXPERIMENT-8

SIMULATING ARP PROTOCOLS

AIM: To write a java program for simulating ARP protocols using TCP

PROCEDURE:

CLIENT-SIDE

1. Start the program
2. Using socket connection is established between client and server.
3. Get the IP address to be converted into MAC address.
4. Send this IP address to the server.
5. Server returns the MAC address to the client.

SERVER-SIDE

1. Start the program
2. Accept the socket which is created by the client.
3. Server maintains the table in which IP and corresponding MAC addresses are stored.
4. Read the IP address which is sent by the client.
5. Map the IP address with its MAC address and return the MAC address to the client.

SOURCE CODE:

Server

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

class Serverarp
{
```

```

public static void main(String args[])
{
try{
ServerSocket obj=new ServerSocket(139);
Socket obj1=obj.accept();
while(true)
{
DataInputStream din=new DataInputStream(obj1.getInputStream());
DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());

String str=din.readLine();

String ip[]={"165.165.80.80","165.165.79.1"};
String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};
for(int i=0;i<ip.length;i++)
{
if(str.equals(ip[i]))
{
dout.writeBytes(mac[i]+"\\n"); break;
}
}
obj.close();
}
}
catch(Exception e)
{
System.out.println(e);
} } }

```

Client

```

import java.io.*;

import java.net.*;

import java.util.*;

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

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

Socket clsct=new Socket("127.0.0.1",139);

DataInputStream din=new DataInputStream(clsct.getInputStream());

DataOutputStream dout=new DataOutputStream(clsct.getOutputStream());
System.out.println("Enter the Logical address(IP):");

String str1=in.readLine(); dout.writeBytes(str1+'\n');

String str=din.readLine();

System.out.println("The Physical Address is: "+str);

clsct.close();}

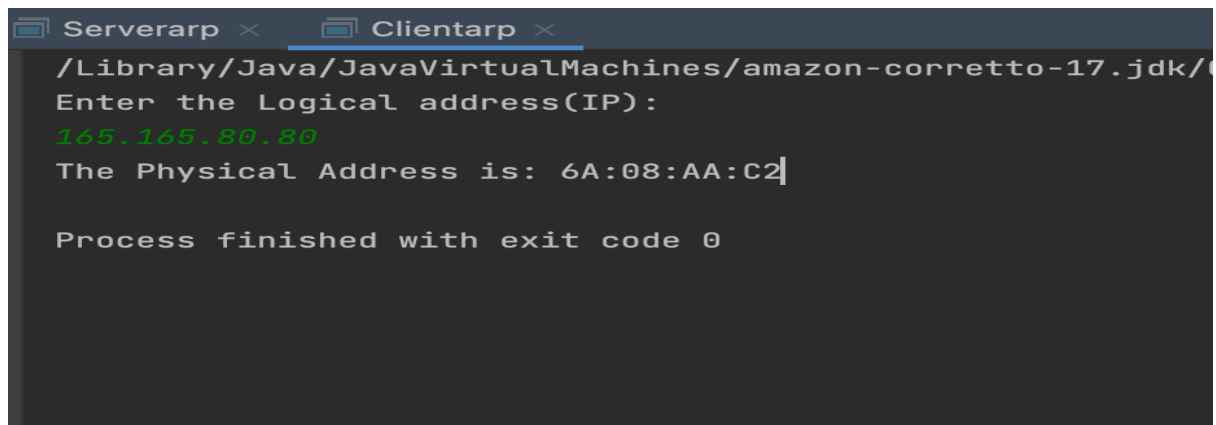
catch (Exception e){

System.out.println(e);

}}
}

```

OUTPUT:



```

Serverarp x Clientarp x
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/
Enter the Logical address(IP):
165.165.80.80
The Physical Address is: 6A:08:AA:C2

Process finished with exit code 0

```

RESULT: Java program for simulating ARP protocols using TCP have been executed successfully.

EXPERIMENT-9

Program for Reverse Address Resolution Protocol (RARP) using UDP

AIM: To write a java program for simulating RARP protocols using DCP

PROCEDURE:

CLIENT SIDE

1. Start the program
2. using datagram sockets UDP function is established.
2. Get the MAC address to be converted into IP address.
3. Send this MAC address to server.
4. Server returns the IP address to client..

SERVER SIDE

1. Start the program
2. Accept the socket which is created by the client.
3. Server maintains the table in which IP and corresponding MAC addresses are stored.
4. Read the IP address which is sent by the client.
5. Map the IP address with its MAC address and return the MAC address to the client.

SOURCE CODE:

Server

```
import java.io.*;

import java.net.*;

import java.util.*;

class Serverrarp12

{
```

```

public static void main(String args[]){
try{
DatagramSocket server=new DatagramSocket(1309);

while(true)

{
byte[] sendbyte=new byte[1024];

byte[] receivebyte=new byte[1024];

DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);
server.receive(receiver);

String str=new String(receiver.getData());

String s=str.trim();

InetAddress addr=receiver.getAddress();

int port=receiver.getPort();

String ip[]={"165.165.80.80","165.165.79.1"};

String mac[]={"6A:08:AA:C2","8A:BC:E3:FA"};

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

{

if(s.equals(mac[i]))

{

sendbyte=ip[i].getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,port);

server.send(sender); break;

}

}

break;

}

}

```

```

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

```

Client

```

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

class Clientarp12{

public static void main(String args[]){

try{

DatagramSocket client=new DatagramSocket();

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

byte[] sendbyte=new byte[1024];

byte[] receivebyte=new byte[1024];

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the Physical address (MAC):")

String str=in.readLine();

sendbyte=str.getBytes();

DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,1309);

client.send(sender);

DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);

client.receive(receiver);

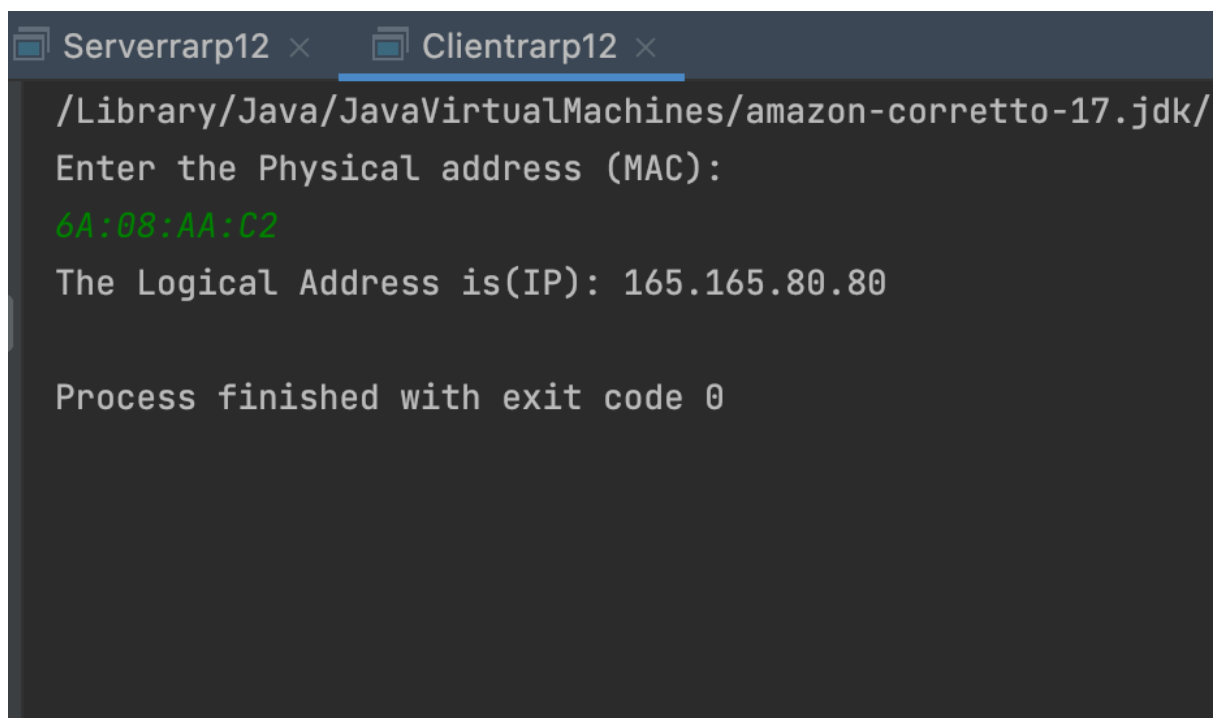
String s=new String(receiver.getData());

System.out.println("The Logical Address is(IP): "+s.trim());

```

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

OUTPUT:



The screenshot shows a Java IDE with two tabs: 'Serverarp12' and 'Clientarp12'. The 'Clientarp12' tab is active and displays the following output:

```
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/  
Enter the Physical address (MAC):  
6A:08:AA:C2  
The Logical Address is(IP): 165.165.80.80  
  
Process finished with exit code 0
```

RESULT: Java program for simulating RARP protocols using DCP has been implemented successfully

EXPERIMENT-10

APPLICATIONS USING TCP SOCKETS

Echo client and Echo server

AIM: To write a java program for application using TCP Sockets Links

PROCEDURE:

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

SOURCE CODE:

Server

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

public class EServer
{
    public static void main(String args[])
    {
        ServerSocket s=null;
        String line;
```

```
DataInputStream is;

PrintStream ps;

Socket c=null;

try
{
s=new ServerSocket(9000);
}
catch(IOException e)
{
}

try
{
System.out.println(e);

c=s.accept();

is=new DataInputStream(c.getInputStream());
ps=new PrintStream(c.getOutputStream());

while(true)
{
line=is.readLine();
ps.println(line);
}
}
catch(IOException e)
{
System.out.println(e);
}
}
```

```
}
```

Client

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

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

```
public class EClient
```

```
{
```

```
public static void main(String arg[])
```

```
{
```

```
Socket c=null;
```

```
String line;
```

```
DataInputStream is,is1;
```

```
PrintStream os;
```

```
try
```

```
{
```

```
InetAddress ia = InetAddress.getLocalHost();
```

```
c=new Socket(ia,9000);
```

```
}
```

```
catch(IOException e)
```

```
{
```

```
}
```

```
try
```

```
{
```

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

```
os=new PrintStream(c.getOutputStream());
```

```
is=new DataInputStream(System.in);
```

```
is1=new DataInputStream(c.getInputStream());
```

```
while(true)
```

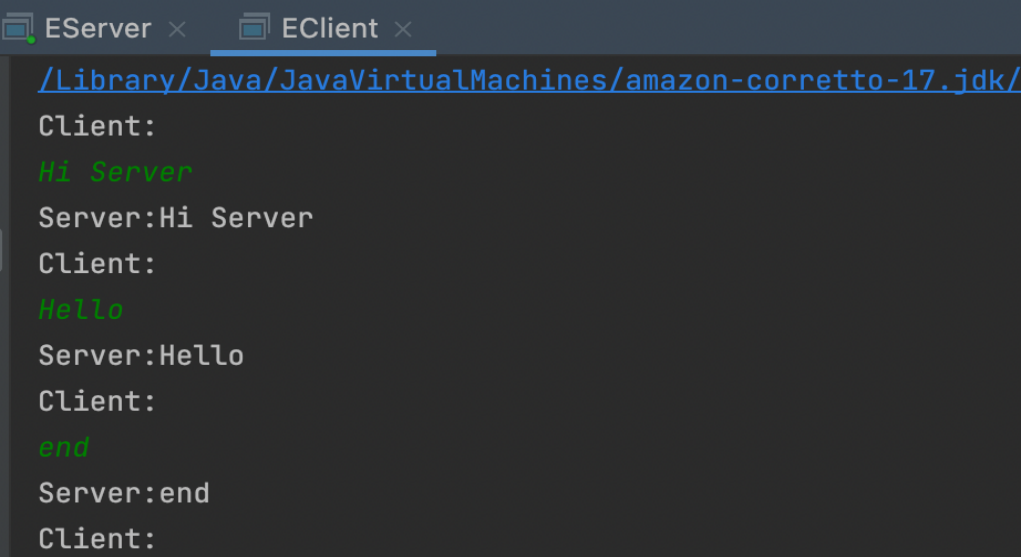
```

{
System.out.println("Client:");
line=is.readLine();
os.println(line);
System.out.println("Server:" + is1.readLine());

}
catch(IOException e)
{
System.out.println("Socket Closed!");
}
} }

```

OUTPUT:



The screenshot shows a Java IDE with two tabs: EServer and EClient. The EClient tab is active, displaying the output of the program. The output shows a series of messages exchanged between the client and the server. The client sends "Hi Server", "Hello", and "end". The server responds with "Hi Server", "Hello", and "end". The client then sends "end" again, and the server responds with "end". The client then sends "end" again, and the server responds with "end". The client then sends "end" again, and the server responds with "end".

```

/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/
Client:
Hi Server
Server:Hi Server
Client:
Hello
Server:Hello
Client:
end
Server:end
Client:
end
Server:end
Client:
end
Server:end

```

RESULT: Java program for application using TCP Sockets Links is implemented successfully

EXPERIMENT-11

CHAT

AIM: Write a Program client –server application for chat using UDP Sockets

PROCEDURE:

1. Start the program.
2. Run UDPserver.java
3. Enter message from server
4. Run UDPclient.java
5. Enter message from client.
6. Stop the program

SOURCE CODE:

Server

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

class UDPserver
{
    public static DatagramSocket ds;
    public static byte buffer[]=new byte[1024];
    public static int clientport=789,serverport=790;
    public static void main(String args[])throws Exception
```

```

{
ds=new DatagramSocket(clientport);
System.out.println("press ctrl+c to quit the program");
BufferedReader dis=new BufferedReader(new InputStreamReader(System.in));
InetAddress ia=InetAddress.getLocalHost();
while(true)
{
DatagramPacket p=new DatagramPacket(buffer,buffer.length);
ds.receive(p);
String psx=new String(p.getData(),0,p.getLength());
System.out.println("Client:" + psx); System.out.println("Server:");
String str=dis.readLine();
if(str.equals("end"))
break;
buffer=str.getBytes();
ds.send(new DatagramPacket(buffer,str.length(),ia,serverport));
}}

```

Client

```

import java .io.*;
import java.net.*;
class UDPclient{
public static DatagramSocket ds;
public static int clientport=789,serverport=790;
public static void main(String args[])throws Exception
{ byte buffer[]=new byte[1024];
ds=new DatagramSocket(serverport);
BufferedReader dis=new BufferedReader(new InputStreamReader(System.in));

```

```
System.out.println("server waiting");

InetAddress ia=InetAddress.getLocalHost();

while(true)

{ System.out.println("Client:");

String str=dis.readLine();

if(str.equals("end"))

break;

buffer=str.getBytes();

ds.send(new DatagramPacket(buffer,str.length(),ia,clientport));

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

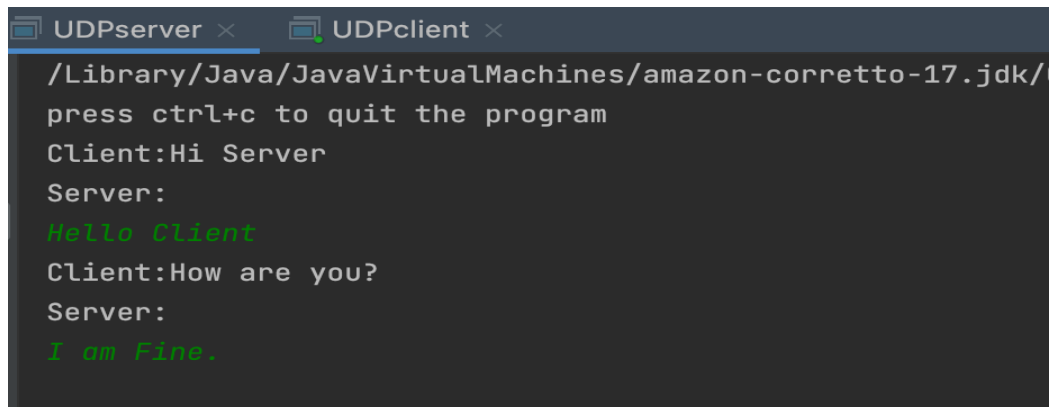
ds.receive(p);

String psx=new String(p.getData(),0,p.getLength());

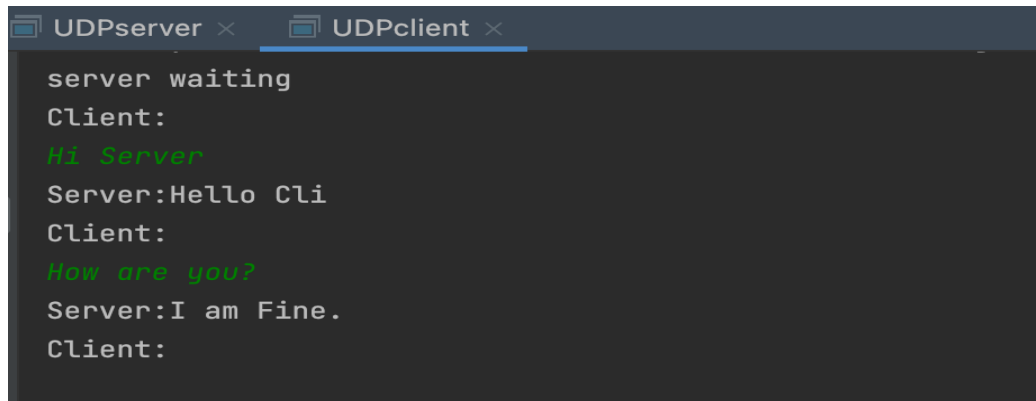
System.out.println("Server:" + psx);

}}}
```

OUTPUT:



```
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/
press ctrl+c to quit the program
Client:Hi Server
Server:
Hello Client
Client:How are you?
Server:
I am Fine.
```



```
UDPServer x  UDPClient x
server waiting
Client:
Hi Server
Server:Hello Cli
Client:
How are you?
Server:I am Fine.
Client:
```

RESULT: Thus, the client –server application for chat using UDP Sockets is implemented successfully.

EXPERIMENT-12

FILE TRANSFER

AIM: Write a program for transferring of files from client to server

PROCEDURE:

1. Start the program.
2. Run Serverfile.java
3. Enter some text and store it the sample.text
4. Run ClientFile and write the file name you want to view
5. Text will be retrieved.
6. Stop the program

SOURCE CODE:

Client

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



```

import java.util.*;

class Clientfile

{ public static void main(String args[])

{

Try

{

BufferedReader in=new BufferedReader(new

InputStreamReader(System.in)); Socket clsct=new Socket("&quot;127.0.0.1&quot;;139);

DataInputStream din=new DataInputStream(clsct.getInputStream());

DataOutputStream dout=new DataOutputStream(clsct.getOutputStream());

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

String str=in.readLine();

dout.writeBytes(str+"\n"); System.out.println("Enter the new file name:");

String str2=in.readLine();

String str1,ss;

FileWriter f=new

FileWriter(str2); char buffer[];

while(true)

{ str1=din.readLine();

if(str1.equals("-1")) break;

System.out.println(str1);

buffer=new char[str1.length()];

str1.getChars(0,str1.length(),buffer,0);

f.write(buffer);

}

f.close();

clsct.close();

```

```

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

```

Server

```

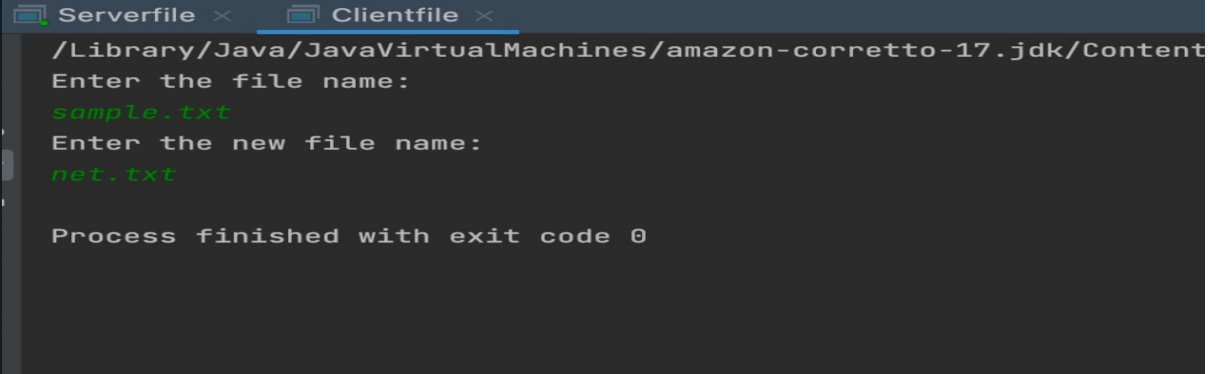
import java.io.*;
import java.net.*;
import java.util.*;
class Serverfile
{ public static void main(String args[])
{
Try
{ ServerSocket obj=new ServerSocket(139);
while(true)
{ Socket obj1=obj.accept();
DataInputStream din=new DataInputStream(obj1.getInputStream());
DataOutputStream dout=new DataOutputStream(obj1.getOutputStream());
String str=din.readLine();
FileReader f=new FileReader(str);
BufferedReader b=new BufferedReader(f);
String s;
while((s=b.readLine())!=null) {
System.out.println(s);
dout.writeBytes(s+"\n");
f.close();
dout.writeBytes("-1\n");} }
catch(Exception e)

```

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

}}
```

OUTPUT:



```
/Library/Java/JavaVirtualMachines/amazon-corretto-17.jdk/Content
Enter the file name:
sample.txt
Enter the new file name:
net.txt

Process finished with exit code 0
```

RESULT: Thus the transferring of files from client to server is implemented successfully.

EXPERIMENT-13

Simulation of DNS using UDP sockets

AIM: To write a java program for DNS application

PROCEDURE:

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.

SOURCE CODE:

Server

```
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();

    }}}

```

Client

```

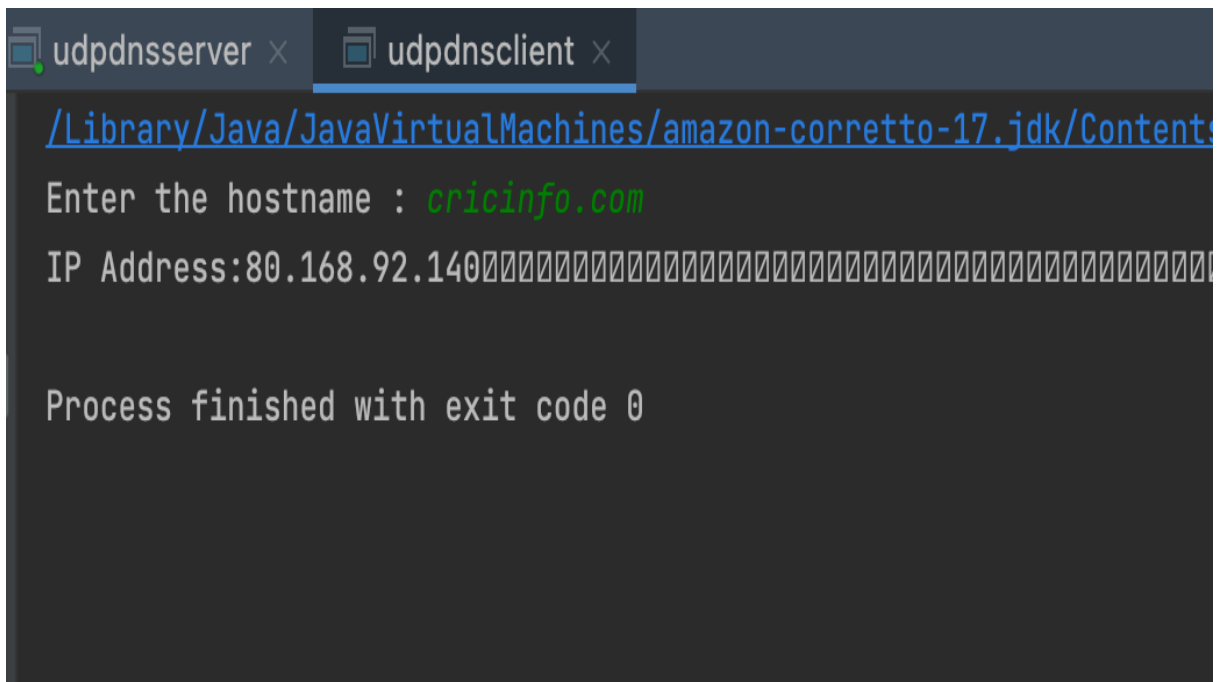
import java.io.*;
import java.net.*;

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);

```

```
System.out.println("IP Address: " + modified); clientsocket.close(); }}
```



[illegible]

```

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;

    *           Constructor

    *           @param add

    */

    public SNMPManager(String add)

    {

        address = add;

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

            /**

            *           Port 161 is used for Read and Other operations

            *           Port 162 is used for the trap generation */

```



```

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

/**
 * OID - .1.3.6.1.2.1.1.1.0 => SysDec
 * OID - .1.3.6.1.2.1.1.5.0 => SysName
 * => MIB explorer will be usefull here, as discussed in previous article */

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

}

/**
 *          get any answers because the communication is asynchronous
 *          and the listen() method listens for answers.
 *          @throws IOException
 */

private void start() throws IOException {
    TransportMapping transport = new DefaultUdpTransportMapping();
    snmp = new Snmp(transport);

    // Do not forget this line! transport.listen();
}

/**
 *          Method which takes a single OID and returns the response from the agent as a
    String.
 *          @param oid
 *          @return
 *          @throws IOException
 */

public String getAsString(OID oid) throws IOException {
    ResponseEvent event = get(new OID[] { oid });

```

```

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

/**
 *          This method is capable of handling multiple OIDs
 *
 *          @param oids
 *
 *          @return
 *
 *          @throws IOException
 */

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");
}

/**
 *          This method returns a Target, which contains information about
 *
 *          where the data should be fetched and how.
 *
 *          @return
 */

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;  
}}
```

OUTPUT:

Hardware: x86 Family 6 Model 23 Stepping 10 AT/AT COMPATIBLE – Software:

Windows 2000 Version 5.1 (Build 2600 Multiprocessor Free)

RESULT: B.SNMP has been implemented Successfully