CS-8581 COMPUTER NETWORKS LABORATORY

Dharaneeshwar P – 1115181040286

CSE -A

Experiment 1:

Netstat

```
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\Dharaneeshwar\Lab Programs\CN>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    127.0.0.1:5000         DTL-Zenbook:55472      TIME_WAIT
  TCP    192.168.1.5:49434      40.90.189.152:https    ESTABLISHED
  TCP    192.168.1.5:55347      40.119.249.228:https   ESTABLISHED
  TCP    192.168.1.5:55356      whatsapp-cdn-shv-01-maa2:https  ESTABLISHED
  TCP    192.168.1.5:55357      172.253.118.188:5228   ESTABLISHED
  TCP    192.168.1.5:55359      lb-140-82-114-25-iad:https  ESTABLISHED
  TCP    192.168.1.5:55398      104.244.42.200:https   ESTABLISHED
  TCP    192.168.1.5:55469      server-13-33-179-97:https  ESTABLISHED
  TCP    192.168.1.5:55473      52.109.56.34:https     TIME_WAIT

D:\Dharaneeshwar\Lab Programs\CN>netstat -e
Interface Statistics

                           Received            Sent

Bytes                      63852330        31686252
Unicast packets              136536           82092
Non-unicast packets             930           31908
Discards                          0               0
Errors                            0               0
Unknown protocols                 0

D:\Dharaneeshwar\Lab Programs\CN>
```

Ipconfig

```
D:\Dharaneeshwar\Lab Programs\CN>ipconfig

Windows IP Configuration


Wireless LAN adapter Local Area Connection* 11:

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

Wireless LAN adapter Local Area Connection* 12:

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

Wireless LAN adapter Wi-Fi:

   Connection-specific DNS Suffix  . : domain.name
   Link-local IPv6 Address . . . . . : fe80::74f7:d7f4:321:2748%18
   IPv4 Address. . . . . . . . . . . : 192.168.1.5
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::c6e9:aff:fe3f:56d0%18
                                       192.168.1.1

D:\Dharaneeshwar\Lab Programs\CN>
```

Ns-lookup

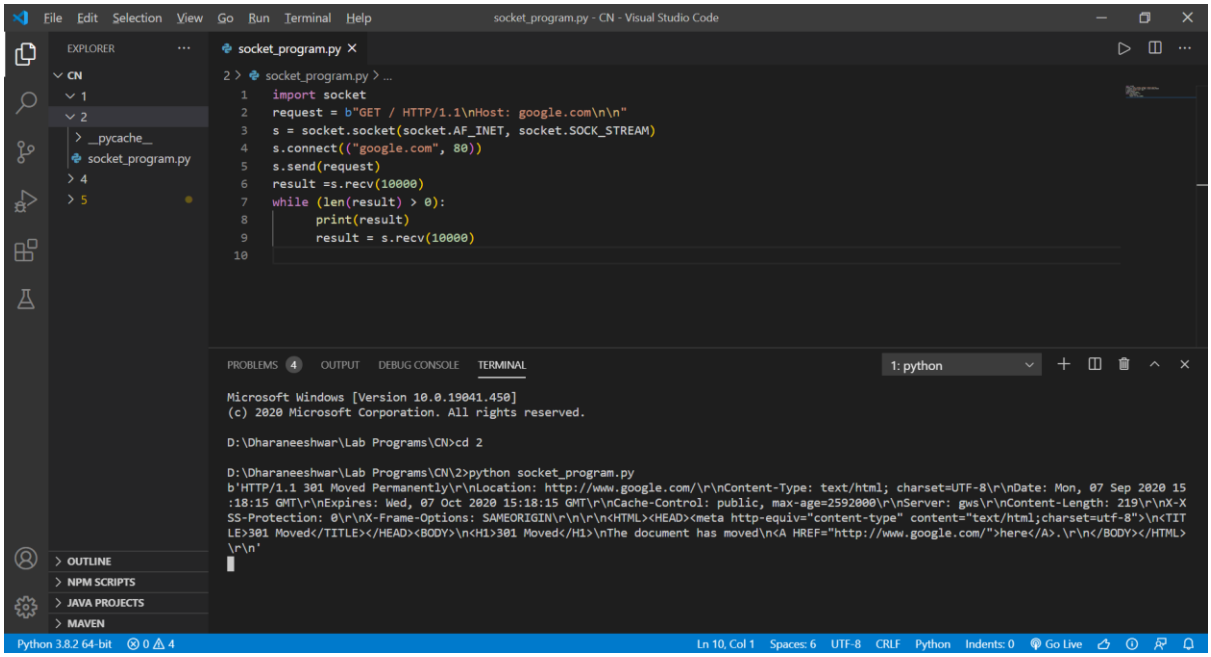
```
D:\Dharaneeshwar\Lab Programs\CN>nslookup www.google.com
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
Name:    www.google.com
Addresses:  2404:6800:4007:804::2004
          172.217.163.196


D:\Dharaneeshwar\Lab Programs\CN>nslookup www.daranip.com
Server:  dns.google
Address:  8.8.8.8

Non-authoritative answer:
Name:    www.daranip.com
Address:  185.199.110.153


D:\Dharaneeshwar\Lab Programs\CN>
```

Traceroute

```
D:\Dharaneeshwar\Lab Programs\CN>tracert www.daranip.com

Tracing route to www.daranip.com [185.199.110.153]
over a maximum of 30 hops:

  1     2 ms     2 ms     2 ms  192.168.1.1
  2    21 ms     *       19 ms  117.193.240.1
  3    22 ms     *       21 ms  218.248.161.106
  4    21 ms    21 ms    20 ms  117.216.207.214
  5    20 ms    20 ms    20 ms  117.216.207.215
  6     *        *        *     Request timed out.
  7     *        *        *     Request timed out.
  8     *        *        *     Request timed out.
  9     *        *        *     Request timed out.
 10    21 ms    21 ms    21 ms  1.7.160.109
 11    93 ms    92 ms    90 ms  185.199.110.153

Trace complete.

D:\Dharaneeshwar\Lab Programs\CN>
```

Experiment 2:

```
import socket
request = b"GET / HTTP/1.1\nHost: google.com\n\n"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("google.com", 80))
s.send(request)
result =s.recv(10000)
while (len(result) > 0):
    print(result,end='\n')
    result = s.recv(10000)
```



EXPERIMENT 3:

(a) TCP SOCKET APPLICATION -ECHO CLIENT AND ECHO SERVER

SERVER:

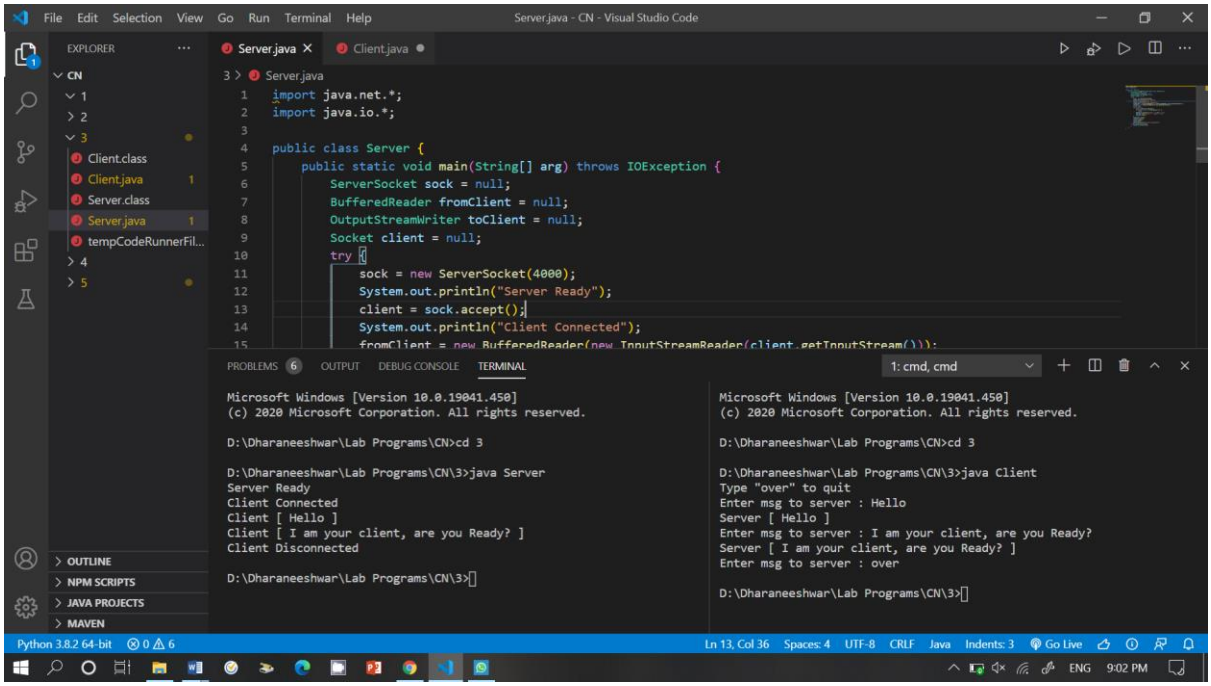
```
import java.net.*;
import java.io.*;

public class Server {
    public static void main(String[] arg) throws IOException {
        ServerSocket sock = null;
        BufferedReader fromClient = null;
        OutputStreamWriter toClient = null;
        Socket client = null;
        try {
            sock = new ServerSocket(4000);
            System.out.println("Server Ready");
            client = sock.accept();
            System.out.println("Client Connected");
            fromClient = new BufferedReader(new
InputStreamReader(client.getInputStream()));
            toClient = new OutputStreamWriter(client.getOutputStream());
```

```
            String line;
            while (true) {
                line = fromClient.readLine();
                if ((line == null) || line.equals("over"))
                    break;
                System.out.println("Client [ " + line + " ]");
                toClient.write("Server [ " + line + " ]\n");
                toClient.flush();
            }
            fromClient.close();
            toClient.close();
            client.close();
            sock.close();
            System.out.println("Client Disconnected");
        } catch (IOException ioe) {
            System.err.println(ioe);
        }
    }
}
```

CLIENT

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

public class Client {
    public static void main(String[] args) throws IOException {
        BufferedReader fromServer = null, fromUser = null;
        PrintWriter toServer = null;
        Socket sock = null;
        try {
            if (args.length == 0)
                sock = new Socket(InetAddress.getLocalHost(), 4000);
            else
```

```
            sock = new Socket(InetAddress.getByName(args[0]), 4000);
        fromServer = new BufferedReader(new
InputStreamReader(sock.getInputStream()));
        fromUser = new BufferedReader(new InputStreamReader(System.in));
        toServer = new PrintWriter(sock.getOutputStream(), true);
        String Usrmsg, Srvmsg;
        System.out.println("Type \"over\" to quit");
        while (true) {
            System.out.print("Enter msg to server : ");
            Usrmsg = fromUser.readLine();
            if (Usrmsg == null || Usrmsg.equals("over")) {
                toServer.println("over");
                break;
            } else
                toServer.println(Usrmsg);
            Srvmsg = fromServer.readLine();
            System.out.println(Srvmsg);
        }
        fromUser.close();
        fromServer.close();
        toServer.close();
        sock.close();
    } catch (IOException ioe) {
        System.err.println(ioe);
    }
  }
}
```



3(b) CHAT APPLICATION USING TCP SOCKETS

SERVER:

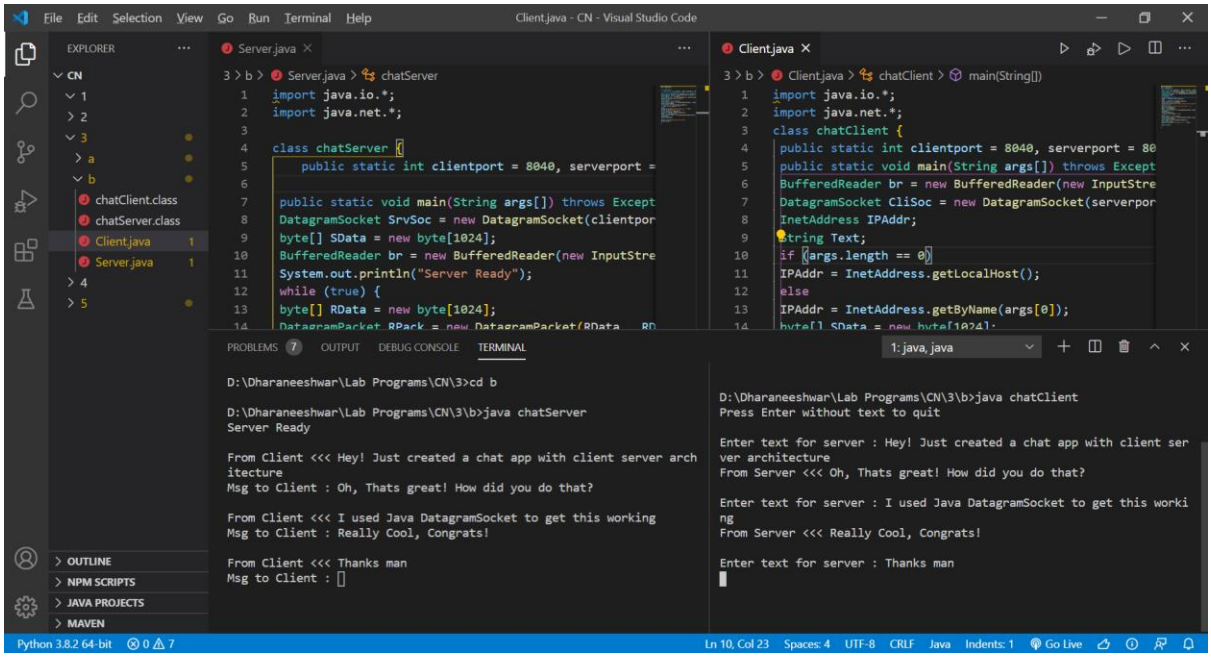
```
import java.io.*;
import java.net.*;
class chatServer {
 public static int clientport = 8040, serverport = 8050;
 public static void main(String args[]) throws Exception {
 DatagramSocket SrvSoc = new DatagramSocket(client port);
 byte[] SData = new byte[1024];
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
 System.out.println("Server Ready");
 while (true) {
 byte[] RData = new byte[1024];
 DatagramPacket RPack = new DatagramPacket(RData , RData.length);
 SrvSoc.receive(RPack);
 String Text = new String(RPack.getData());
 if (Text.trim().length() == 0)
 break;
 System.out.println("\nFrom Client <<< " + Text);
```

```
 System.out.print("Msg to Client : ");
 String srvmsg = br.readLine();
 InetAddress IPAddr = RPack.getAddress();
 SData = srvmsg.getBytes();
 DatagramPacket SPack = new DatagramPacket(SData,SData.length,IPAddr,
 serverport);
 SrvSoc.send(SPack);
 }
 System.out.println("\nClient Quits\n");
 SrvSoc.close();
 }
}
```

CLIENT

```
import java.io.*;
import java.net.*;
class chatClient {
 public static int clientport = 8040, serverport = 8050;
 public static void main(String args[]) throws Exception {
 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
 DatagramSocket CliSoc = new DatagramSocket(serverport);
 InetAddress IPAddr;
 String Text;
 if (args.length == 0)
 IPAddr = InetAddress.getLocalHost();
 else
 IPAddr = InetAddress.getByName(args[0]);
 byte[] SData = new byte[1024];
 System.out.println("Press Enter without text to quit");
 while (true) {
 System.out.print("\nEnter text for server : ");
 Text = br.readLine();
```

```
SData = Text.getBytes();
DatagramPacket SPack = new DatagramPacket(SData, SData.length, IPAddr,
clientport);
CliSoc.send(SPack);
if (Text.trim().length() == 0)
break;
byte[] RData = new byte[1024];
DatagramPacket RPack = new DatagramPacket(RData, RData.length);
CliSoc.receive(RPack);
String Echo = new String(RPack.getData());
Echo = Echo.trim();
System.out.println("From Server <<< " + Echo);
}
CliSoc.close();
}
}
```



3(C) FILE TRANSFER USING TCP SOCKETS

SERVER:

```
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
```

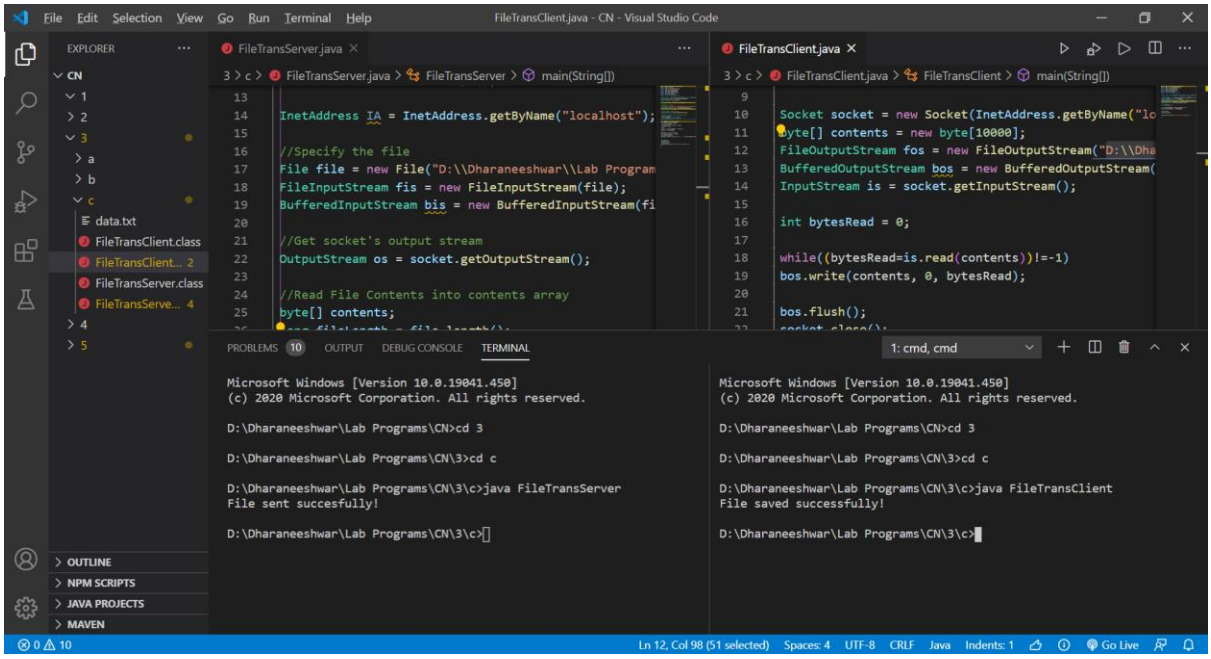
```
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
public class fileTransServer {

 public static void main(String[] args) throws Exception {
 ServerSocket sock = new ServerSocket(5000);
 Socket socket = sock.accept();

 InetAddress IA = InetAddress.getByName("localhost");

 //Specify the file
 File file = new File("C:\\Users\\Divya\\Documents\\labb.txt");
 FileInputStream fis = new FileInputStream(file);
 BufferedInputStream bis = new BufferedInputStream(fis);

 //Get socket's output stream
 OutputStream os = socket.getOutputStream();

 //Read File Contents into contents array
 byte[] contents;
 long fileLength = file.length();
 long current = 0;

 long start = System.nanoTime();
 while(current!=fileLength){
 int size = 10000;
 if(fileLength - current >= size)
 current += size;
 else{
 size = (int)(fileLength - current);
```

```
 current = fileLength;
 }
 contents = new byte[size];
 bis.read(contents, 0, size);
 os.write(contents);
 System.out.print("Sending file ... "+(current*100)/fileLength+"% complete!");
 }

 os.flush();
 socket.close();
 sock.close();
 System.out.println("File sent succesfully!");
 }
}
```

CLIENT

```
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.Socket;
public class fileTransClient {

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

 Socket socket = new Socket(InetAddress.getByName("localhost"), 5000);
 byte[] contents = new byte[10000];
 FileOutputStream fos = new
FileOutputStream("C:\\Users\\Divya\\Documents\\labb.txt");
 BufferedOutputStream bos = new BufferedOutputStream(fos);
 InputStream is = socket.getInputStream();
```

```
 int bytesRead = 0;

 while((bytesRead=is.read(contents))!=-1)
 bos.write(contents, 0, bytesRead);

 bos.flush();
 socket.close();

 System.out.println("File saved successfully!");
 }
}
```



EXPERIMENT 4: SIMULATION OF DNS USING UDP SOCKETS

SERVER:

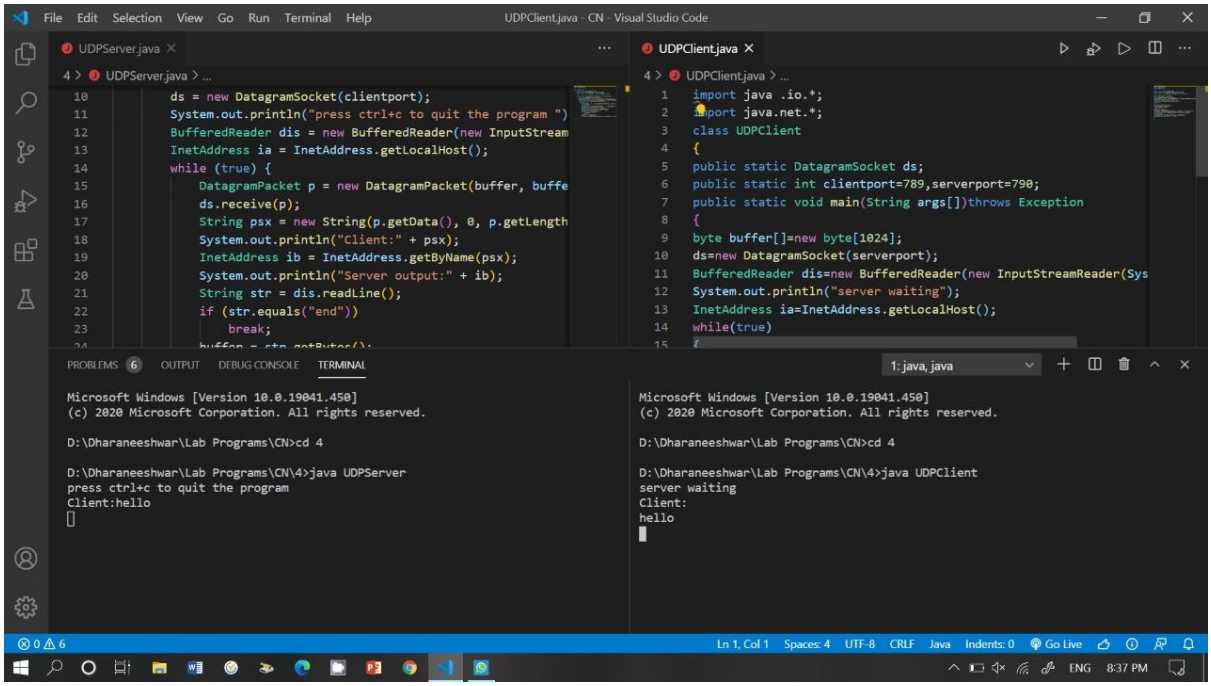
```
import java.io.*;
import java.net.*;
class UDPServer
{
public static DatagramSocket ds;
public static byte buffer[]=new byte[1024];
```

```
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);
InetAddress ib=InetAddress.getByName(psx);
System.out.println("Server output:"+ib);
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);
}
}
}
```

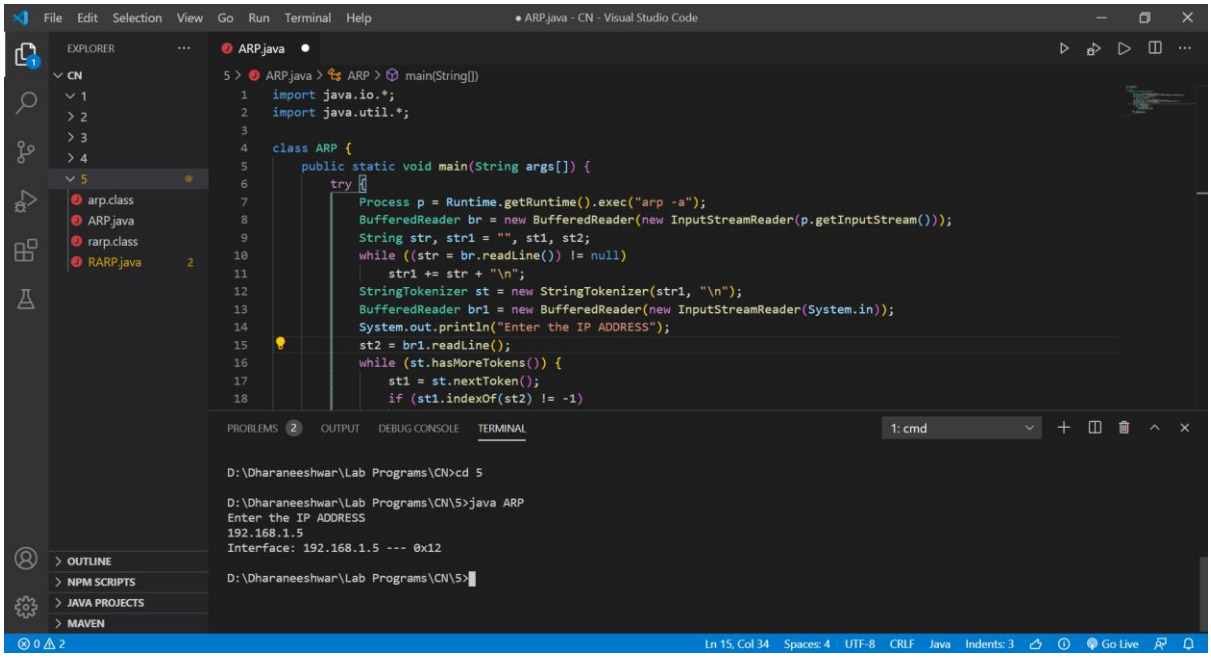


EXPERIMENT 5: SIMULATING ARP PROTOCOL USING JAVA

ARP

```
import java.net.*;

import java.io.*;

import java.lang.Object;

import java.util.*;

class arp

{

public static void main(String args[])

 {

try

 {

 Process p=Runtime.getRuntime().exec("arp -a");

BufferedReader br=new BufferedReader(new
InputStreamReader(p.getInputStream()));

 String str,str1="",st1,st2;

while((str=br.readLine())!=null)

 str1+=str+"\n";
```

```
 StringTokenizer st=new StringTokenizer(str1,"\n");
 BufferedReader br1=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the IP ADDRESS");
 st2=br1.readLine();
while(st.hasMoreTokens())
 {
 st1=st.nextToken();
if(st1.indexOf(st2)!=-1)
System.out.println(st1);
 }
 }
catch(Exception e)
 {
e.printStackTrace();
 }
 }
}
```



RARP

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

```
import java.io.*;
import java.lang.Object;
import java.util.*;
class rarp
{
public static void main(String args[])
 {
try
 {
 Process p=Runtime.getRuntime().exec("arp -a");
 BufferedReader br=new BufferedReader(new
InputStreamReader(p.getInputStream()));
 String str,str1="",st1,st2;
while((str=br.readLine())!=null)
 str1+=str+"\n";
 StringTokenizer st=new StringTokenizer(str1,"\n");
BufferedReader br1=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the physical 48-bit ADDR ENTER THE PHYSICAL 48BIT
ADDRESS ");
 st2=br1.readLine();
while(st.hasMoreTokens())
 {
 st1=st.nextToken();
if(st1.indexOf(st2)!=-1)
System.out.println(st1);
 }
 }
catch(Exception E)
 {
E.printStackTrace();
 } } }
```

RARP.java - CN - Visual Studio Code

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

class RARP {
    public static void main(String args[]) {
        try {
            Process p = Runtime.getRuntime().exec("arp -a");
            BufferedReader br = new BufferedReader(new InputStreamReader(p.getInputStream()));
            String str, str1 = "", st1, st2;
            while ((str = br.readLine()) != null)
                str1 += str + "\n";
            StringTokenizer st = new StringTokenizer(str1, "\n");
            BufferedReader br1 = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter the physical 48-bit ADDR ENTER THE PHYSICAL 48BIT ADDRESS ");
            st2 = br1.readLine();
            while (st.hasMoreTokens()) {
                st1 = st.nextToken();
                if (st1.indexOf(st2) != -1)
```

```
Microsoft Windows [Version 10.0.19041.450]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\Dharaneeshwar\Lab Programs\CN>cd 5

D:\Dharaneeshwar\Lab Programs\CN\5>java RARP
Enter the physical 48-bit ADDR ENTER THE PHYSICAL 48BIT ADDRESS
DC-8B-28-9B-8F-1C

D:\Dharaneeshwar\Lab Programs\CN\5>
```