

1. TO IDENTIFY WELL KNOWN PORTS

This program will identify the ports which are active also shows host name and address of active ports.

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

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

```
class ports
```

```
{
```

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

```
    {
```

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

```
        String host = h.getHostNamE();
```

```
        String haddr = h.getHostAddress();
```

```
        System.out.println("\nHost Name :: " + host);
```

```
        System.out.println("\nHost Address :: " + haddr);
```

```
        for (int i = 0; i < 1024; i++)
```

```
        {
```

```
            socket s = null;
```

```
            try
```

```
            {
```

```
                s = new Socket (host, i);
```

```
                System.out.println("port " + i + "Active");
```

```
            }
```

```
            catch (SocketException se)
```

```
            {
```

```
            }
```

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

Output :

| | | |
|------|---|----------|
| post | 0 | Inactive |
| post | 1 | Inactive |
| post | 2 | Inactive |
| post | 3 | Inactive |
| post | 4 | Inactive |
| post | 5 | Inactive |
| post | 6 | Inactive |
| post | 7 | Inactive |
| post | 8 | Inactive |
| post | 9 | Inactive |

2. PROGRAMS FOR CHAT APPLICATION

2a) one to one chat Application :-

* SERVER SIDE PROGRAM *

This program is used to perform one to one chatting. it needs two java programs i.e., server program and client program.

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

```
public class chats extends Thread
```

```
{
```

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

```
    {
```

```
        try
```

```
        {
```

```
            ServerSocket ss = new ServerSocket (111);
```

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

```
            Socket s = ss.accept();
```

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

```
            BufferedReader br = new BufferedReader (new InputStreamReader  
                (s.getInputStream()));
```

```
            PrintWriter pw = new PrintWriter (new OutputStreamWriter  
                (s.getOutputStream()));
```

```

while(true)
{
    String msg = br.readLine();
    System.out.println("from client: "+msg);
    if(msg.equals("quit"))
    {
        System.out.println("disabled");
        break;
    }
    System.out.println("server ");
    msg = in.readLine();
    pw.println(msg);
    pw.flush();
}
}
catch(IOException e)
{
    System.out.println("error");
}
}
}

```

* CLIENT SIDE PROGRAM *

This is a client program to run with chats.java, It reads client data sends to server and accepts server data to print on the screen

```

import java.io.*;
import java.net.*;
import java.lang.*;
public class chatc
{
    public static void main (String args[])
    {
        try
        {
            BufferedReader in = new BufferedReader(new InputStreamReader
                (System.in));

            Socket s = new Socket ("localhost", 1111);
            BufferedReader br = new BufferedReader (new InputStreamReader
                (s.getInputStream()));
            PrintWriter pw = new PrintWriter(s.getOutputStream());
            while (true)
            {
                System.out.println ("client");
                String msg = in.readLine ();
                pw.println (msg);
                pw.flush ();
                String from = br.readLine ();
                System.out.println ("from server = " + from);
                if (from.equals ("quit"))
                {

```

```

System.out.println("Disconnect");
System.exit(0);
}
}
}
catch (IOException e)
{
    System.out.println("ending");
}
}
}
}

```

output:

C:\nwlab> java chats

server ready

from client: Tell me the best book for java basics

server

Balaguruswamy or the complete reference

from client: that's nice

server

ok then bye

from client: thank u bye

server

C:\nwlab>

2b) MANY TO MANY CHATTING

* SERVER SIDE PROGRAM *

This program perform multi chatting:
many servers can chat with many clients.
It is server program uses array of
socket to connect.

```
import java.io.* ;
import java.net.* ;
public class msServer
{
    public static Socket s[] = new Socket [10];
    public static String user [] = new String [10];
    public static String int total ;
    public static void main (String a [])
    {
        int i = 0 ;
        try
        {
            ServerSocket ss = new ServerSocket (118) ;
            server socket
            while (true)
            {
                s [i] = ss. accept ();
```

```

BufferedReader br = new BufferedReader(new Input-
StreamReader(s[i].getInputStream()));
String msg = br.readLine();
user[i] = msg;
System.out.println(msg + " converted");
try
{
    reqHandler req = new reqHandler(s[i], i);
    total = i;
    i++;
    Thread t = new Thread(req);
    t.start();
}
catch (Exception e)
{
    System.out.println(e);
}
}
catch (Exception e)
{
    System.out.println(e);
}
}
}

```

class reqhandler implements Runnable

{

public int n;

public Socket s;

public reqhandler(Socket soc, int i)

{

s = soc;

n = i;

}

public void run()

{

String msg = " ";

BufferedReader br;

PrintWriter pw;

try

{

while (true)

{

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

msg = br.readLine();

if (msg.equals("quit"))

msrvr.total--;

else

System.out.println (msrvr.user[n] + " → " + msg);

```

if (mserver.total == -1)
{
    System.out.println("server Disconnected..");
    System.exit(0);
}
for (int k = 0; k <= mserver.total; k++)
{
    if (!mserver.users[k].equals(mserver.users[n])
    && (!msg.equals("quit")))
    {
        pw = new PrintWriter(new OutputStreamWriter
        (mserver.s[k].getOutputStream()));
        pw.println(mserver.users[n] + ": " + msg + "\n");
        pw.flush();
    }
}
}
catch (Exception e)
{ }
}
}

```

* CLIENT SIDE PROGRAM *

client program to run with mserver.
java, it connects to the localhost with
port no 118 and sends message to server
to which it is connected

```
import java.io.*;
import java.net.*;
public class mclient
{
    public static void main (String a [])
    {
        BufferedReader in ;
        PrintWriter pw ;
        try
        {
            Socket s = new Socket ("localhost", 118);
            System.out.println ("Enter name:");
            in = new BufferedReader (new InputStreamReader (System.in));
            String msg = in.readLine ();
            pw = new PrintWriter (new OutputStreamWriter (s.getOutputStream ()));
            pw.println (msg + "\n");
            pw.flush ();
        }
    }
}
```

```

while (true)
{
    xaddata xd = new xaddata(s);
    Thread t = new Thread(xd);
    t.start();
    msg = in.readLine();
    if(msg.equals("quit"))
    {
        System.exit(0);
    }
    pw.println(msg);
    pw.flush();
}
catch (Exception e)
{
    System.out.println(e);
}
}
}

```

class xaddata implements Runnable

```

{
    public Socket s;
    public xaddata(Socket s)
    {
        this.s = s;
    }
}

```

```

public void run()
{
    BufferedReader br;
    try
    {
        while(true)
        {
            br = new BufferedReader(new InputStreamReader
                (s.getInputStream()));

            String msg = br.readLine();
            System.out.println(msg);
        }
    }
    catch (Exception e)
    {
        System.out.println(e);
    }
}
}
}

```

output :

```

C:\nwlabs> java mserver
Miley connected
Scott connected

```

output :-

```

C:\nwlabs> java mclient
Enter name :
Scott

```

output :

```

C:\nwlabs> java mclient
Enter name :
Miley

```

3. DATA RETRIEVAL FROM REMOTE DATABASE

Problem description:-

At the remote database a server listens for client connections. This server accepts SQL queries from the client, executes it on the database and sends the response to the client.

Remote database:-

A database to which a connection is made using a database link which is connected to a local database.

SERVER SIDE PROGRAM

server program that sends database requested by the client.

```
import java.sql.*;
```

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

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

```
class db server
```

```
{
```

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

```
{
```

```
connection con;
```

```
Result rs;
```

```
statement s;
```

```
ResultSet Metadata sm = null;
```

```
server Socket ss;
```

```

Socket soc ;
BufferedReader br ;
PrintWriter pw ;
s = con. createStatement ( ) ;
ss = new ServerSocket ( 8504 ) ;
while ( true )
{
try
{
soc = ss. accept ( ) ;
System. out. println ( " connected ..... " ) ;
pw = new PrintWriter ( soc. getOutputStream ( ) ) ;
br = new BufferedReader ( new InputStreamReader
( soc. getInputStream ( ) ) ) ;

String query = br. readLine ( ) ;
rs = s. executeQuery ( query ) ;
the records in rs
if ( rs != null )
rm = rs. getMetaData ( ) ;
int cnt = rm. getColumnCount ( ) ;
for ( int i = 1 ; i < cnt ; i ++ )
pw. print ( rm. getColumnName ( i ) + " | " ) ;
names in output stream
pw. println ( "\n" ) ;
pw. flush ( ) ;
int ct = 0 ;

```

```

while (xs.next ())
{
    for (int i=1; i <= cnt; i++)
        pw.println(xs.getString(i) + " ");
    pw.println("\n");
    pw.flush();
    ++ct;
}
pw.println("\n\n" + ct + " Row(s) selected");
pw.flush();
pw.println("end#");
pw.flush();
pw.close();
bx.close();
soc.close();
System.out.println("service complete.....");
}
catch (Exception e)
{
}
}
}

```

Client side Program

This program connects to database and request the data base from the server in the form of SQL query. To connect to the server it asks to enter server id and then query.

```
import java.sql.*;
import java.net.*;
import java.io.*;
class dbclient
{
    public static void main(String args[]) throws Exception
    {
        Socket soc;
        BufferedReader br, keyin;
        PrintWriter pw;
        String name, data;
        keyin = new BufferedReader(new InputStreamReader
            (System.in));
        System.out.println("Enter Remote server ID:");
        name = keyin.readLine();
        System.out.println("Enter UR Query");
        while(true)
        {
            System.out.println("SQL >");
            String sql = keyin.readLine();
            if(sql.equals("quit"))
                break;
        }
    }
}
```

```

soc = new Socket (net Address. get ByName (name), 8504);
pw = new PrintWriter (soc. get outputStream());
br = new BufferedReader (new InputStreamReader
(soc. get Input Stream()));

pw.println (391);
pw.flush ();
while (true)
{
data = br.readLine ();
if (data. equals ("end #"))
break;
System.out.println (data);
}
}
}
}

```

Output :-

C:\nwlab > java dbserver

connected -----

Service complete -----

C:\nwlab > java dbclient

Enter Remote server ID: Local host

Enter UR Query

sql > select * from software - emp

ID | NAME | JOB |

1001 | Vinay | Analyst |

1002 | Nagendra | Programmer |

1003 | Satish | Tester |

1004 | Manikanta | Software Eng. |

4 Row(s) selected

sql >

4. FTP CLIENT SERVER PROGRAM

Problem Description:-

By opening socket connection to our server on one system and sending a file from one system to another.

SERVER SIDE PROGRAM

This program is FTP server which creates a user connection to copy a file from remote location, it takes the source and target file path from the client.

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

public class ftp server
{
    ServerSocket ss = new ServerSocket(9000);
    System.out.println("server started");
    try
    {
        while (true)
        {
            Socket s = ss.accept();
            DataInputStream di = new
            DataInputStream(s.getInputStream());
            PrintStream p = new PrintStream(s.getOutputStream());
            String st = di.readLine();
            String temp = st;
```

```

temp = "file:// " + st ;
URL U = new URL (temp);
URL connection UC = U.openConnection();
FileInputStream dil = new FileInputStream(st);
int ch ;
while (ch = dil.read()) != -1)
{
System.out.println ((char)ch);
p.write(ch);
}
s.close();
}
}
catch (Exception e)
{
}
}
}
}

```

CLIENT SIDE PROGRAM .

This program is ftp client which connects to ftp server to get the file copied from source to target remote location.

```

import java.io.*;
import java.net.*;
Public class ftp client
{

```

```

Public Static void main(String args[]) throws Exception
{
    Socket s = new Socket("localhost", 9000);
    BufferedReader di = new BufferedReader(new Input
        StreamReader(s.getInputStream()));
    InputStream in = s.getInputStream();
    DataInputStream di = new DataInputStream(System.in);
    PrintStream p = new PrintStream(s.getOutputStream());
    System.out.println("Enter absolute path")
    String st = di.readLine();
    p.println(st);
    System.out.println("enter destination");
    String file = di.readLine();
    FileOutputStream fos = new FileOutputStream(file);
    int ch;
    while ((ch = di.read()) != -1)
    {
        System.out.println((char)ch);
        fos.write(ch);
    }
}
}

```

Output :-

C:\nwlab > java ftp server

server started

CPP: C Plus Plus

it is an object oriented prog.

its features:

Data abstraction

encapsulation

inheritance

Polymorphism

C:\nwlab >

Output :-

C:\nwlab > java ftpclient

Enter absolute path

C:\nwlab *CPP.txt

enter destination

C:\nwlab\CPPCOPY.txt

CPP: C Plus Plus

it is an object oriented prog

its features:

Data Abstraction

encapsulation

inheritance

Polymorphism

C:\nwlab >

5. TRIVIAL FILE TRANSFER PROTOCOL

SERVER SIDE PROGRAM

This TFTP server program copies the source file to target file using datagram packets the TFTPserver class is extended to thread class.

```
import java.io.* ;
import java.net.* ;
class TFTPMS extends Thread
{
    Thread t ;
    DatagramSocket ds ;
    TFTPMS (DatagramSocket ds)
    {
        try
        {
            t = new Thread (this) ;
            this.ds = ds ;
            t.start() ;
        }
        catch (Exception e)
        {
        }
    }
    public void run()
    {
```

```

try
{
    int pos = 0 ;
    int buff_size = 2000 ;
    byte buff [] = new byte [buff_size] ;
    DatagramPacket dp = new DatagramPacket(buff, buff.length);
    ds.receive(dp);
    String filename = new String(dp.getData(), 0, dp.getLength());

    FileInputStream fis = new FileInputStream(filename);
    int c ;
    while ((c = fis.read()) != -1)
    {
        buff[pos++] = (byte) c ;
        if (pos % 2000 == 0)
        {
            ds.send(new DatagramPacket(buff, pos, InetAddress.
                getLocalHost(), 50)) ;
        }
    }
    catch (Exception e)
    {
    }
}
}

public class tftp server
{
    public static void main(String args[]) throws Exception
    {
        DatagramSocket ds ;
    }
}

```

```

System.out.println("server read");
ds = new DatagramSocket(69);
FTPMS ms = new FTPMS(ds);
}
}

```

CLIENT SIDE PROGRAM

This FTP client program responds to FTP server it enters absolute path and target file path which is sent to server.

```

import java.io.*;
import java.net.*;
class FTPClient
{
    public static void main (String args [])
    {
        FTPC = new FTPC ();
    }
}
class FTPC
{
    DatagramSocket ds;
    int buff-size = 2000;
    byte buff [] = new byte [buff-size];
    FTPC ()
    {
        .
        .
    }
}

```

```

ds = new DatagramSocket(50);
}
catch (Exception e)
{
System.exit(0);
}
DataInputStream in = new DataInputStream(System.in);
try
{
System.out.println("Enter absolute path of file");
String str = in.readLine();
int pos = str.length();
byte buf[] = new byte[pos];
for (int i = 0; i < pos; i++)
buf[i] = (byte) str.charAt(i);
ds.send(new DatagramPacket(buf, pos, InetAddress.
getLocalHost(), 69));
System.out.println("enter the name to be saved");
String stl = in.readLine();
FileOutputStream fos = new FileOutputStream(stl);
while (true)
{
DatagramPacket dp = new DatagramPacket(buf, buf.
length);
ds.receive(dp);
String file = new String(dp.getData(), 0, dp.getLength());
}
}

```

```
int g = file.length();  
for (int j = 0; j < g; j++)  
fos.write(("char") file.charAt(j));  
}  
}  
catch (Exception e)  
{  
}  
}  
}
```

output :-

```
C:\nwlab > java tftp server  
server ready  
C:\nwlab > _
```

output :-

```
C:\nwlab > java tftp client  
Enter absolute path of file:  
C:\nwlab \ CPP.txt  
Enter the name to be saved  
C:\nwlab \ CPP1.txt  
C:\nwlab >
```

6. SIMULATION OF TELNET (Remote Login)

Program :

It is a gui program to perform telnet. it is a menu driven program which has options connect - to connect, dis connect and exit, the object such as text area, textbox, scroll pane, menu and buttons of swing class are used.

```
import java.io.* ;
import java.net.* ;
import javax.swing.* ;
import java.awt.* ;
import java.awt.event ;

public class telnet extends JFrame implements
Action Listener, KeyListener
{
    public static JTextArea ta ;
    JMenuBar jm ;
    JMenu comMenu ;
    JMenuItem conHost, discon, exit ;
    boolean edited = false ;
    Socket soc ;
    BufferedReader br ;
    PrintWriter pw ;
    public static telnet t ;
    String command ;
```

```

Public Telnet ( )
{
Set Title ("Telnet ");
Set size (400, 400);
Command = new String ( );
ta = new JTextArea ( );
ta.addKeyListener (This);
getContentPane ( ).add (new JScrollPane (ta));
JM = new JMenuBar ( );
set JMenuBar (JM);
Commenu = new JMenu ("connect");
JM.add (Commenu);
conhost = new JMenuItem ("connect");
conhost.addActionListener (this);
discon = new JMenuItem ("Disconnect");
discon.addActionListener (this);
exit = new JMenuItem ("Exit");
exit.addActionListener (this);
Commenu.set Mnemonic ('c');
Commenu.add (conhost);
Commenu.add (discon);
Commenu.add (exit);
setVisible (true);
new Dialog ( );
}
Public void key pressed (KeyEvent ke)
{

```

```

}
public void keyReleased (key Event ke)
{
}
public void keyTyped (key Event ke)
{
if (ke. getkeychar() == keyEvent.VK-ENTER)
{
System.out.println (command);
pw.println (command);
command = " ";
}
else
if (ke. getkeychar() != keyEvent.VK-SHIFT)
command = command + ke. getkeychar();
}
public void actionPerformed (ActionEvent ae)
{
object ob = ae. getSource();
if (ob == exit)
System.exit(0);
else
if (ob == conhost)
new conDialog();
if (ob == discon)
if (!(soc == null))
{

```

```
System.out.println("connection closed");
```

```
try
```

```
{
```

```
    soc.close();
```

```
    soc = null;
```

```
}
```

```
catch (Exception e)
```

```
{
```

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

```
}
```

```
}
```

```
}
```

```
void makeConnection()
```

```
{
```

```
    try
```

```
    {
```

```
        soc = new
```

```
        Socket (condialog.getHostText().trim(), Integer.
```

```
        parseInt (condialog.getPortText().trim()));
```

```
        JOptionPane.showMessageDialog (null, "connection  
        established");
```

```
        System.out.println ("connection established");
```

```
        br = new BufferedReader (new InputStreamReader (soc.get-  
        InputStream()));
```

```
        pw = new PrintWriter (soc.getOutputStream(), true);
```

```
        ta.append (reply + "\n");
```

```
        ta.setCursorPosition (reply.length());
```

```
        new Thread ().start();
```

```
    }
```

```
    catch (Exception e)
```

```

{
System.out.println(e);
OptionPane.showMessageDialog(null, "connection to
host lost");
}
}
class ReadThread extends Thread
{
public void run()
{
try
{
int off = 0;
while (true)
{
String reply = br.readLine();
if (reply == null) System.exit(1);
ta.append(reply + "\n");
int lc = ta.getLineCount();
off = ta.getLineStartOffset(lc);
ta.setCaretPosition(off);
}
}
catch (Exception e) {
System.out.println(e);
OptionPane.showMessageDialog(null, "connection TO
Host lost");
}
}
}
}

```

```
Public static void main (String a[])
```

```
{  
    t = new ternet();
```

```
    }  
}
```

```
class CDialog implements ActionListener
```

```
{  
    JDialog jd ;
```

```
    Public static JTextField host, port ;
```

```
    JButton connect, cancel ;
```

```
    Public CDialog ()
```

```
{
```

```
    jd = new JDialog ();
```

```
    host = new JTextField (40);
```

```
    port = new JTextField (40);
```

```
    connect = new JButton ("CONNECT");
```

```
    cancel = new JButton ("CANCEL");
```

```
    connect.addActionListener (this);
```

```
    cancel.addActionListener (this);
```

```
    jd.setBounds (200, 300, 300, 150);
```

```
    jd.getContentPane().setLayout (new GridLayout (3, 2));
```

```
    jd.getContentPane().add (new JLabel ("Remote Host:"));
```

```
    jd.getContentPane().add (host);
```

```
    jd.getContentPane().add (new JLabel ("Port Number:"));
```

```
    jd.getContentPane().add (port);
```

```
    jd.getContentPane().add (connect);
```

```
    jd.getContentPane().add (cancel);
```

```
    jd.setVisible (true);
```

```
}
```

```
Public void action performed (ActionEvent ae)
{
object ob = ae.getSource();
if (ob == cancel)
    jd.dispose();
else
if (ob == connect)
{
jd.dispose();
telnet.t. Makeconnection();
}
}
}
```

OUTPUT

```
C:\nwlab> javac talent.java
C:\nwlab> java telnet
connection established
this is talent program.
```

7. HYPER TEXT TRANSFER PROTOCOL

server side program

A server side program for implementing HTTP protocol

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

public class htps
{
    public static void main(String args[]) throws Exception
    {
        ServerSocket s1 = new ServerSocket(2000) ;
        Socket con1 = s1.accept() ;
        System.out.println("connected to server") ;
        BufferedReader commands from client = new BufferedReader
            (new InputStreamReader(con1.getInputStream()));
        PrintWriter response to client = new
            PrintWriter(con1.getOutputStream(), true);
        BufferedReader input = new BufferedReader(new Input
            StreamReader(System.in));

        int choice;
        do
        {
            choice = Integer.parseInt(commands.readLine());
            String file;
            byte line[] = null;
            File f;
            switch (choice)
```

```
{ resource
```

case 1 :-

```
System.out.println("1. Head");  
file = commandsFromClient.readLine();  
f = new File(file);  
int index = file.lastIndexOf(".");  
String type = file.substring(index + 1);  
responseToClient.println(type);  
long length = f.length();  
responseToClient.println(length);  
break;
```

case 2 :-

```
System.out.println("2 POST");  
file = commandsFromClient.readLine();  
System.out.println("MESSAGE POSTED FROM CLIENT:");  
System.out.println(file);  
break;
```

case 3 :-

```
System.out.println("3. GET");  
file = commandsFromClient.readLine();  
FileInputStream fis = new FileInputStream(file);  
while (fis.available() != 0)  
{  
    if (fis.available() < 1024)  
        line = new byte[fis.available()];  
    fis.read(line);  
}
```

```
file = new String (line);  
response To client. println (file);  
}  
response To client. println ("***");  
fis. close ();  
break ;
```

case 4 :-

```
System.out.println ("4. DELETE");  
file = command From client. readLine ();  
f = new File (file);  
f.delete ();  
break ;  
default. System.out.println ("5. EXIT");  
System.exit (0);  
}  
}  
while (choice <= 4);  
con1. close ();  
s1. close ();  
}  
}
```

client side program

A client program for implementing
http

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

public class htpc
{
    public static void main(String args[]) throws Exception
    {
        Socket con1 = new Socket("localhost", 2000);
        BufferedReader responseFrom server = new BufferedReader
            (new InputStreamReader(con1.getInputStream()));
        PrintWriter commands to server = new PrintWriter(con1.getOutputStream(), true);
        BufferedReader input = new BufferedReader(new InputStreamReader
            (System.in));
        System.out.println("CONNECTED TO SERVER");
        int choice;
        do
        {
            System.out.println("COMMANDS");
            System.out.println("1-HEAD 2-POST 3-GET 4-DELETE 5-EXIT");
            System.out.println("ENTER YOUR CHOICE:");
            choice = Integer.parseInt(input.readLine());
            byte line[] = null;
        }
    }
}
```

```
Switch (choice)
```

```
{
```

```
case 1 :
```

```
    commands TO server. println("1");  
    System.out.println("Enter file Name to get the  
                           Header");  
    file = input.readLine();  
    commands TO server. println(file);  
    String type = responseFromServer.readLine();  
    String length = responseFromServer.readLine();  
    System.out.println("FILE: " + file + "TYPE: " + type +  
                       "LENGTH + length);
```

```
break;
```

```
case 2 :-
```

```
    commands TO server. println("2");  
    System.out.println("ENTER TEXT TO POST:");  
    file = input.readLine();  
    commands TO server. println(file);  
break;
```

```
case 3 :-
```

```
    commands TO server. println("3");  
    System.out.println("ENTER FILE NAME TO GET:");  
    file = input.readLine();  
    File outputStream fos = new FileOutputStream  
                           (file);
```

```
while (true)
```

```
{
```

```

String s = response From server. readLine();
if (s. equals ("***"))
break;
int count = s. length();
if (count < 1024)
line = new byte [count];
else
line = new byte [1024];
line = s. get Bytes ();
fos. write (line);
}
fos. close ();
break;

```

case 4 :-

```

    commands To server. println ("4");
System. out. println ("ENTER FILE NAME TO DELETE:");
file = input. readLine();
commands To server. println (file);
break;
default : commands To server. println ("5");
    System. exit (0);
}
}
while (choice <= 4);
con1. close ();
}
}

```

OUTPUT :-

C:\nwlab > java https

connected to server

2. POST

MESSAGE POSTED FROM CLIENT :

welcome to java http program

1. HEAD

2. EXIT

C:\nwlab >

OUTPUT :

C:\nwlab > java httpc

CONNECTED TO SERVER

COMMANDS

1. HEAD 2. POSTS 3. GET 4. DELETE 5. EXIT

ENTER TEXT YOUR CHOICE :

2

ENTER TEXT TO POST :

welcome to java http program

COMMANDS

1. HEAD 2. POSTS 3. GET 4. DELETE 5. EXIT

ENTER YOUR CHOICE : | ENTER YOUR CHOICE :

1

5

ENTER FILE NAME TO GET THE HEADER :

cpp.txt

8. MAIL CLIENTS - POST OFFICE PROTOCOL (POP) & SIMPLE MAIL TRANSFER PROTOCOL (SMTP)

8.1. MAIL CLIENT - POP (COMMAND PROMPT)

POP mail

```
import java.io.* ;
import java.net.* ;
import java.util.* ;
public class POP Mail
{
    public static void main (String args [])
    {
        try
        {
            socket connect to server = new Socket ("127.0.0.1", 110);
            BufferedReader is from server = new BufferedReader
                (new InputStreamReader (connect to server. getInputStream()));
            PrintWriter OS to server = new
                PrintWriter (connect to server. getOutputStream(), true);
            BufferedReader br = new BufferedReader (new InputStreamReader
                (System.in));

            OS to server. println ("user " + "user1");
            System.out.println ("USER RESPONSE IS: " + is from
                server. readLine ());

            OS to server. println ("Pass " + "user1");
            System.out.println ("PASSWORD RESPONSE IS: " + is from
                server. readLine ());
        }
    }
}
```

```

System.out.println(is From server.readLine());
while(true)
{
System.out.println("ENTER THE POP COMMAND:");
String c = br.readLine();
if(c.length() == 0) break;
OS TO server.println(c);
String s = is From server.readLine();
while(!s.equals("."))
{
System.out.println(s);
s = is From server.readLine();
}
}
}
connect TO server.close();
}
catch (Exception e)
{
System.out.println("ERROR: " + e);
}
}
}
}

```

8.2. MAIL CLIENT - SMTP (COMMAND PROMPT)

SMTP client

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

public class SMTP client
{
    public static void main (String args [])
    {
        Socket s ;
        PrintWriter pw ;
        BufferedReader br, from kb ;
        try
        {
            s = new Socket ("localhost", 25) ;
            br = new BufferedReader (new InputStreamReader
                (s.getOutputStream()));
            pw = new PrintWriter (s.getOutputStream(), true);
            from kb = new BufferedReader (new InputStreamReader
                (System.in));

            int ch ;
            String msg = null ;
            do
            {
                System.out.println ("SMTP COMMANDS");
                System.out.println ("1. HELO");
```

```

System.out.println("2. MAILFROM USER NAME");
System.out.println("3. RCPT TO USERNAME");
System.out.println("4. DATA & SEND");
System.out.println("5. EXIT");
System.out.println("ENTER YOUR CHOICE!");
ch = Integer.parseInt(from kb.readLine());
switch(ch)
{

```

case 1 :-

```

    pw.println("HELO");
    break;

```

case 2 :-

```

    System.out.println("ENTER senders address");
    msg = from kb.readLine();
    pw.println("Mail FROM: " + msg); break;

```

case 3 :-

```

    System.out.println("Enter receivers address");
    msg = from kb.readLine();
    pw.println(msg); p("RCPT TO: " + msg);
    break;

```

case 4 :-

```

    System.out.println("Enter data to send");
    msg = from kb.readLine(); pw.println("DATA");
    pw.println(msg);
    pw.println(".");

```

```
Pw. println("QUIT");  
break;  
default : system.exit(0);  
}  
}  
while (ch < 5);  
}  
catch (Exception e)  
{  
    System.out.println(e);  
}  
}  
}
```