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
exce
{
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 :

```
Port   0   Inactive
Port   1   Inactive
Port   2   Inactive
Port   3   Inactive
Port   4   Inactive
Port   5   Inactive
Port   6   Inactive
Port   7   Inactive
Port   8   Inactive
Port   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 InputStream-
                                    Reader(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 PROGRM *

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", 111);
BufferedReader br = new BufferedReader (new InputStream
                                        Reader(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 byc
from client: thank q byc
 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 mserver
{
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 + "connected");
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"))
  mserver.total--;
else
System.out.println(mserver.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.user[k].equals(mserver.user[n])
 && (!msg.equals("quit")))
 {
 pw=new PrintWriter(new OutputStreamWriter
        (mserver.s[k].getOutputStream()));
 pw.println(mserver.user[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 InputStreamRea-
                        der(System.in));

String msg =in.readLine();
pw =new printWriter (new OutPutStreamWriter
                    (s.getOutPutStream()));

pw.println (msg + "\n");
pw.flush();
```

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

class xreaddata implements Runnable
{
 Public Socket s;
 Public xreaddata(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:\nwlab> java mserver
Miller connected
Scott connected
```

output:-

```
C:\nwlab> java mclient
Enter name:
Scott
```

output:

```
C:\nwlab> java mclient
Enter name:
Miller
```

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;
ResultSetMetaData rm=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 qry = br.readline();
 rs = s.executeQuery(qry);
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 outputstream
pw.println("\n");
pw.flush();
 int ct = 0;
```

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

Clint side Program

This program connects to dbserver 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 BufferReader (new Input StreamReader
                        (soc.get Input stream()));

pw.println(sql);
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 | softwar Eng. |
4 Row(s) selected
sql >
```

4. FTP CLIENT SERVER PROGRAM

Problem Description :-

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

SERVER SIDE PROGRAM

This program is ftp server which creates a url 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();
Data InputStream id = 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);
URLConnection 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 ftpClient
{
```

```
Public Static void main(String args[])throws Exception
{
Socket s=new Socket("localhost",9000);
BufferedReader di=new BufferedReader(new InputStreamReader(s.getInputStream()));
InputStream in=s.getinputStream();
DataInputStream di=newDataInputStream(System.in);
PrintStream p=new PrintStream(s.get OutputStream());
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 File outputStream(file);
int ch;
while(cch=di.read()!=-1)
{
System.out.println((char)ch);
fos.write(ch);
}
}
}
```

output :-

```
C:\nw lab > java ftp server
server started
CPP : cp plus plus
it is an object oriented prag.
its features:
Data abstraction
encapsulation
inheritance
Polymorphism
C:\nwlab >
```

output :-

```
C:\nw lab > 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;
Datagram Socket ds;
tftpms (Datagram Socket 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.recive(dP);
String filename = new String(dP.getData(), 0, dP.get
                                                length());
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 servey
{
Public static void main(String args[]) throws Exception
{
DatagramSocket ds;
```

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

CLIENT SIDE PROGRAM

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

```
import java.io.*;
import java.net.*;
class tftpClient
{
Public static void main (String args[])
{
tftpc = new tftpc();
 }
}
classtftpc
{
DatagramSocket ds;
int buff_size = 2000;
byte buff[] = new byte[buff_size];
tftpc()
{
  try
{
```

```
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 stt =in.readline();
FileOutputStream fos =new FileOutputStream(stt);
while(true)
{
DatagramPacket dP =new DatagramPacket(buff,buff.
                                    length);

ds.recive(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:\nw lab > java tftp server
server ready
c:\nwlab>_
```

output :-

```
c:\nwlab > java tftp client
Enter absoult path of file:
c:\nwlab\cpp.txt
Enter the name to be saved
c:\nw lab\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, disconnect 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 talent extends Jframe implements
Action Listener, keylistener
{
Public Static JTextArea ta;
JMenuBar jm;
Jmenu conmenu;
JmenuItem conhost, discon, exit;
boolean edited = false;
Socket soc;
BufferedReader br;
PrintWriter PW;
Public Static telnet t;
String command;
```

```
Public telnet ( )
{
Set Title ("Telnet ");
 Setsize (400, 400) ;
Command = new String ( ) ;
ta =new JTexTArea( ) ;
ta. addkeylistener (This ) ;
get contentPane ( ). add (new J Scrollpane(ta)) ;
JM =new JMenu Bar( ) ;
set JMenu Bar (JM) ;
Commenu =new JMenu ("connect") ;
JM. add (commenu) ;
conhost =new JMenuItem ("connect") ;
conhost. addActionListener (this) ;
discon = new JMenuItem ("Disconnect") ;
discon. add Actionlistener (this) ;
exit = new JMenuItem ("Exit") ;
exit. add Actionlistener (this) ;
commenu. set Mnemonic ('c') ;
commenu. add (conhost) ;
commenu. add (discon) ;
commenu. add (exit) ;
 setvisible (true) ;
new condialog ( ) ;
 }
Public void key pressed (keyEvent ke)
{
```

```
}
Public Void keyReleased (key Event ke)
{
}
Public void key Typed (key Event ke)
{
if (ke. get key char( ) = = key Event. Vk_ENTER)
{
System. out. println (command);
pw. println(command);
command = " ";
}
else
if(ke. getkeychar( ) ! =keyEvent. Vk_SHIFT)
command = command +ke. get keychar( );
}
Public void actionperfoymed (ActionEvent ae)
{
object ob =ae. get source( );
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.host.getText().trim(), integer.
        parseInt(condialog.port.getText().trim()));
  JoptionPane.showMessageDialog(null,"connection
                                    established");
System.out.println("connection established");
br=new BufferedReader(newInputStreamReader(soc.get-
                                    InputStream()));
  pw=new PrintWriter(soc.getOutputStream(),true);
  ta.append(reply+"\n");
  ta.setCaretPosition(reply.length());
  new readthread().start();
  }
 catch(Exception e)
```

```
{
System.out.println(e);
JoptionPane.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.getLine start offset(lc);
ta.setCaret Position(off);
  }
 }
 catch(Exception e);
{
 System.out.println(e);
joptionpane.showMessageDialog(null,"connection To
                                    Host Lost");
   }
  }
}
```

```
Public Static Void main (String a[])
{
 t = new telnet();
  }
}
class condialog implements Actionlistener
{
  JDialog Jd;
Public Static JTextField. host, port;
JButton connect, cancel;
Public candjalog()
{
jd = new JDialog();
host = new JTextField (40);
port = new JTextField (40);
connect = new JButton ("CONNECT");
cancel = new JButton ("CANCEL");
connect.addAction listener (this);
cancel. addActionListener (this);
Jd. getBounds (200, 300, 300, 150);
Jd. getcontentpane(). set Layout(new GridLayout(3,2));
jd. gedcontentPanel(). add (new JLabel ("Remote Host."));
jd. get contentPane(). add (host);
jd. get contentpane(). add (new JLabel ("port Number:"));
jd. get contentPane(). add (port);
jd. get Content Pane(). add (content);
jd. get content pane(). add (cancel);
 jd. set Visible (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 > java c 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.long.*;
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=newBufferedReader
        (newInputStreamReader(con1.getPutstream()));
PrintWriter response to client=new
PrintWriter(con1.getoutputStream(), true);
BufferedReader input=new BufferedReader(newInput
                                StreamReader(system.in));

int choice;
    do
{
choice=integer.parseInt(commandFromclient.readline());
 String file;
byte line[]=null;
 File f;
 Switch(choice)
```

```
{
resource
case 1 :-
        System.out.println(" 1. Head ");
    file = commandsFromClient.readline();
      f = newFile(file);
   int index = file.lastIndex of(". ");
   String type = file.substring (index + 1);
   response TO client.println (type);
   long length = f.length();
   responseToclient.println(length);
   break;

case 2 :-
     System.out.println ("2 POST ");
   file = commands FromClient.readline();
System.out.println("MESSAGE POSTED FROM CLIENT:");
     System.out.println (file);

  break;

case 3 :-
        System.out.println("3.GET ");
   file = commands FromClient.readLine();
  FileInputStream fis = newFileInputStream(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
            (newInputStreamReader(con1.getInputStream()));
PrintWriter commonds TO server = new PrintWriter(con1.get
                                  outputStream(), true);
BufferedReader input = New BufferedReader(newInput StreamReader
                                  (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 = response From Server. 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 File outputStream
                                          (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.getBytes();
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 htps
connected to server
2. POST
MESSAGE POSTED FROM CLIENT :
welcome to java http program
1. HEAD
2. EXIT
c:\nwlab >
```

output :

```
c:\nwlab > java htpc
CONNECTED TO SERVER
COMMANDS
1. HEAD  2. POSTS  3. GET  4. DELETE  5. EXTT
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 fromServer= new BufferedReader
   (new InputStreamReader(connect To server.getInputStream));
PrintWriter OS TO server = new
PrintWriter(connected To server.getOutputStream(), true);
BufferedReader br=new BufferedReader(new InputStreamReader
                                    (System.in));
OS TO server.println("user" + "user1");
System.out.println("USER RESPONE IS:" + is From
                    server.readLine());
OS To server.println("POSS" + "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.read Line();
 }
 }
}
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(newInputStreamReader
                        (s.getOutPutStream()));
pw=new PrintWriter (s.getOutputStream(),true);
from kb=new BufferedReader (new InPut Stream
                        Reader (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=fromkb.readLine();
  pw.println("Mail FROM:"+msg); break;
case 3:-
    System.out.println("Enter recievers address");
    msg=fromkb.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);
 }
 }
}
```