```
// Session 1
//  MyThread2.java
import java.io.FileOutputStream;
import java.io.PrintWriter;

class MyThread2 implements Runnable {
        int name;

        public MyThread2(int ten) {
                this.name = ten;
                System.out.println("Thread " + name + " duoc khoi tao ... !");
        }

        public void run() {
                try {
                         FileOutputStream f = new FileOutputStream("./out." + this.name);
                         PrintWriter pw = new PrintWriter(f);
                         for (int i = 0; i < 100; i++) {
                                String say = "Hello from " + this.name + "-thread";
                                System.out.println(say);
                                pw.println(this.name + "-thread");
                         }
                         pw.flush();
                         f.close();
                } catch (Exception e) {
                        System.out.println("Loi khi truy xuat file.");
                }

        }
        public static void main(String args[]) {
                int nThread = Integer.parseInt(args[0]);
                for (int i = 1; i <= nThread; i++) {
                        MyThread2 th2 = new MyThread2(i);
                        Thread th = new Thread(th2);
                        th.start();
                }

        }

}


// PipedEcho.java
import java.io.*;
public class PipedEcho {
       public static void main(String argv[]) {
               try {
                        PipedOutputStream cwPipe = new PipedOutputStream();
                        PipedInputStream crPipe = new PipedInputStream();
                        PipedOutputStream swPipe = new PipedOutputStream(crPipe);
```

```
            PipedInputStream srPipe = new PipedInputStream(cwPipe);
            PipedEchoServer server = new PipedEchoServer(srPipe, swPipe);
            PipedEchoClient client = new PipedEchoClient(crPipe, cwPipe);
        } catch(IOException ie) {
            System.out.println("Echo server Error: " + ie);
        }
    }
}


// PipedEchoClient.java
import java.io.*;
public class PipedEchoClient extends Thread {
    PipedInputStream readPipe;
    PipedOutputStream writePipe;
    PipedEchoClient(PipedInputStream readPipe, PipedOutputStream writePipe) {
        this.readPipe = readPipe;
        this.writePipe = writePipe;
        System.out.println("Client creation");
        start();
    }
    public void run() {
        while(true) {
            try {
                int ch = System.in.read();
                writePipe.write(ch);
                ch = readPipe.read();
                System.out.print((char)ch);
            }
            catch(IOException ie) {
                System.out.println("Echo server Error: " + ie);
            }
        }
    }
}


// PipedEchoServer.java
import java.io.*;
public class PipedEchoServer extends Thread {
    PipedInputStream readPipe;
    PipedOutputStream writePipe;
    PipedEchoServer(PipedInputStream readPipe, PipedOutputStream writePipe) {
        this.readPipe = readPipe;
        this.writePipe = writePipe;
        System.out.println("Server is starting...");
        start();
    }
    public void run() {
        while(true) {
            try {
                int ch = readPipe.read();
                writePipe.write(ch);
            }
            catch(IOException ie) {
                System.out.println("Echo server Error: " + ie);
            }
        }
    }
}
```

// Session 2

PTCPEchoServer.java

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

public class PTCPEchoServer {
    public final static int defaultPort = 2019;

    public static void main(String args[]) {
        try {
            ServerSocket ss = new ServerSocket(defaultPort);
            System.out.println("server socket is running");
            while (true) {
                try {
                    Socket s = ss.accept();
                    // Tao xu ly
                    RequestProcessing rq = new RequestProcessing(s);
                    rq.start();
                } catch(IOException e) {
                    System.out.println("connection Error: " + e);
                }
            }
        } catch (Exception e) {
            System.out.println("Creat Socket Error: " + e);
        }
    }
}
```

RequestProcessing.java

```
import java.io.*;
import java.net.*;
class RequestProcessing extends Thread
{
    private Socket s;
    public RequestProcessing(Socket s1) {
        s = s1;
    }
    public void run() {
        try {
            OutputStream os = s.getOutputStream();
            InputStream is = s.getInputStream();
            int ch = 0;
            while(true) {
                ch = is.read();
                if(ch == -1) break;
                os.write(ch);
            }
            s.close();
        }
        catch (IOException e) {
            System.err.println("Processing Error: " + e);
        }
    }
}
```

STCPEchoServer.java

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

public class STCPEchoServer {
    public final static int defaultPort = 8080;

    public static void main(String args[]) {
        try {
            ServerSocket ss = new ServerSocket(8080);
            System.out.println("server socket is running");
            while (true) {
                Socket s = ss.accept();
                OutputStream os = s.getOutputStream();
                InputStream is = s.getInputStream();
                int ch = 0;
                while (true) {
                    ch = is.read();
                    if (ch == -1)
                    break;
                    System.out.print((char) ch);
                    os.write(ch);
                }
                s.close();
            }
        } catch (Exception e) {
            System.out.print(e.toString());
        }
    }
}
```

TCPEchoClient.java

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

public class TCPEchoClient {
    public static void main(String args[]) {
        try {
            Socket s = new Socket(args[0], Integer.parseInt(args[1]));
            InputStream is = s.getInputStream();
            OutputStream os = s.getOutputStream();
            while (true) {
                BufferedReader br = new BufferedReader(new
                InputStreamReader(System.in));
                String theString = br.readLine();
                byte[] data = theString.getBytes();
                String quit = new String("quit");
                if (Arrays.equals(quit.getBytes(), data)) {
                    System.out.println("Quit");
                    break;
                }
                for (int i = 0; i < data.length; i++) {
                    os.write(data[i]);
                    int ch = is.read();
                    System.out.print((char) ch);
                }
                System.out.println();
            }
            s.close();
        } catch (Exception e) {
            System.out.print(e.toString());
        }
    }
}
```

```
// Session 3

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

public class chatBotServer {
    public static void main(String[] args) {
        // Handle error
        if(args.length < 1) {
            System.out.println("Port number is required but not provided");
            return;
        }
        // Build hash table
        Hashtable<String, String> question = new Hashtable<String, String>();
        question.put("hi", "Hello");
        question.put("bye", "Goodbye see you later.");

        // New server socket
        try {
            int port = Integer.parseInt(args[0]);
            ServerSocket ss = new ServerSocket(port);
            System.out.println("Server is running on " + port);
            int numConnection = 1;

            while(true) {

                Socket s = ss.accept();
                System.out.println("Connection " + numConnection + ": " + s);
                numConnection++;

                BufferedReader br = new BufferedReader(new
                InputStreamReader(s.getInputStream()));
                PrintWriter pw = new PrintWriter(s.getOutputStream());
                while(true) {
                    // Get quesion form client.
                    String a, q;
                    q = br.readLine();
                    q = q.toLowerCase();
                    boolean find = question.containsKey(q);
                    if(find == true) a = question.get(q);
                    else a = "Chatbot: I am listing you.";
                    // Wrint into client
                    pw.println("Chatbot: " + a);
                    pw.flush();
                    if(q.equals("bye")) break;
                }
                s.close();
            }
        } catch(Exception e) {
            System.out.println(e);
        }
    }
}

// RequestProcessing.java

import java.io.*;
import java.net.*;
class RequestProcessing extends Thread
{
    private Socket s;
    public RequestProcessing(Socket s1) {
        s = s1;
    }
    public void run() {
        try {
            OutputStream os = s.getOutputStream();
            InputStream is = s.getInputStream();
            int ch = 0;
            while(true) {
                ch = is.read();
                if(ch == -1) break;
                os.write(ch);
            }
            s.close();
        }
        catch (IOException e) {
            System.err.println("Processing Error: " + e);
        }
    }
}

// TCPChatClient.java

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

public class TCPChatClient {
    public static void main(String args[]) {
        try {
            // server is listening on port
            if(args.length < 3) {System.out.println("Enter Host Port UserName...");
            return;}
            String userName = args[2];
            Socket s = new Socket(args[0], Integer.parseInt(args[1]));
            InputStream is = s.getInputStream();
            OutputStream os = s.getOutputStream();
            BufferedReader br = new BufferedReader(new
            InputStreamReader(s.getInputStream()));
            PrintWriter pw = new PrintWriter(s.getOutputStream());
            ThreadReader readerFromServer = new ThreadReader(br);
            readerFromServer.start();

            ThreadWritter writterFromClient = new ThreadWritter(pw, userName);
            writterFromClient.start();

        } catch (Exception e) {
            System.out.print(e.toString());
        }
    }
}

 class ThreadReader extends Thread
 {
     private BufferedReader br;
     public ThreadReader(BufferedReader br) {
        this.br = br;
     }
     public void run() {
        try {
            while(true) {
                String str = br.readLine();
                System.out.println(str);
            }
        }
         catch (IOException e) {
            System.out.println("Processing Error: " + e);
         }
     }
 }

class ThreadWritter extends Thread
{
    private PrintWriter pw;
    String userName;
    public ThreadWritter(PrintWriter pw, String userName) {
        this.pw = pw;
        this.userName = userName;
    }
    public void run() {
        try {
            Scanner sc = new Scanner(System.in);
            while(true) {
                String str = sc.nextLine();
                pw.println(userName + ": " + str);
                pw.flush();
            }
        }
        catch (Exception e) {
            System.err.println("Processing Error: " + e);
        }
    }
}

// TCPChatServer.java

import java.io.*;
import java.util.*;
import java.net.*;
// Server class
public class TCPChatServer
{
    public static void main(String[] args) throws IOException
    {
        // server is listening on port
        if(args.length < 2) {System.out.println("Enter PORT UserName..."); return;}
        int port = Integer.valueOf(args[0]).intValue();
        ServerSocket ss = new ServerSocket(port);
        Socket s;
        String userName = args[1];
        try {
            while (true) {
                s = ss.accept();
                System.out.println("New client request received : " + s);

                // obtain input and output streams
                BufferedReader br = new BufferedReader(new
                InputStreamReader(s.getInputStream()));
                PrintWriter pw = new PrintWriter(s.getOutputStream());
                System.out.println("Creating a new handler for this client...");

                // Create a new handler object for handling this request.
                ThreadReader readerFromClient = new ThreadReader(br);
                readerFromClient.start();
                ThreadWritter writterFromServer = new ThreadWritter(pw,
                userName);
                writterFromServer.start();
            }
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}




// ClientHandler clasS
class ThreadReader extends Thread
```

```
{
    private BufferedReader br;
    public ThreadReader(BufferedReader br) {
        this.br = br;
    }
    public void run() {
        try {
            while(true) {
            String str = br.readLine();
            System.out.println(str);
        }
    }
    catch (IOException e) {
        System.err.println("Processing Error: " + e);
    }
    }
}
```

```
class ThreadWritter extends Thread
{
    private PrintWriter pw;
    String userName;
    public ThreadWritter(PrintWriter pw, String userName) {
        this.pw = pw;
        this.userName = userName;
    }
    public void run() {
        try {
            Scanner sc = new Scanner(System.in);
            while(true) {
                String str = sc.nextLine();
                pw.println(userName + ": " + str);
                pw.flush();
            }
        }
        catch (Exception e) {
            System.err.println("Processing Error: " + e);
        }
    }
}
```

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

public class UDPEchoClient {
        public static void main(String[] args) {
                try {
                        if(args.length < 2) {
                                System.out.print("Systax: java UDPClient HostName PORT");
                                return;
                        }
                        int serverPort = Integer.valueOf(args[1]).intValue();
                        // Tao DatagramSocket
                        DatagramSocket ds = new DatagramSocket();
                        // Dia chi server
                        InetAddress server = InetAddress.getByName(args[0]);
                        while(true){
                                BufferedReader br = new BufferedReader(new
                                InputStreamReader(System.in));
                                String theString = br.readLine();
                                // Doi chuoi ra mang bytes
                                byte[] data = theString.getBytes();
                                // Tao goi tin
                                DatagramPacket dp = new DatagramPacket(data, data.length,
                                server, serverPort);
                                ds.send(dp); // gui goi tin sang server
                                // Tao vung dem de nhan goi tin
                                byte[] buffer = new byte[60000];
                                DatagramPacket incoming = new DatagramPacket(buffer,
                                buffer.length);
                                ds.receive(incoming); // cho nhan goi tinh tra loi tu server
                                // Hien thi goi tin ra mang hinh
                                System.out.println(new String(incoming.getData(), 0,
                                incoming.getLength()));
                        }
                } catch(IOException e) {
                        System.err.println(e);
                }
        }
}
```

```
// UDPEchoServer.java
```

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

public class UDPEchoServer {
    public static void main(String[] args) {
        try {
            if(args.length < 1) {System.out.println("Enter PORT..."); return;}
            // Tao socket
            int port = Integer.valueOf(args[0]).intValue();
            DatagramSocket ds = new DatagramSocket(port);
            System.out.println("Created UDP Socket...");
            // Buffer
            byte[] buffer = new byte[60000];
            while(true){
                DatagramPacket in = new DatagramPacket(buffer,
                buffer.length);
                ds.receive(in);
                // Lay du lieu khoi tin nhan
                String str = new String(in.getData(), 0, in.getLength());
                //Tao goi tin goi chua du lieu vua nhan
                DatagramPacket out = new DatagramPacket(str.getBytes(),
                in.getLength(), in.getAddress(), in.getPort());
                ds.send(out);
            }
        } catch(IOException e) {
            System.err.println(e);
        }
    }
}
```

```
// Session 4
SMTPClient.java
import java.io.*;
import java.util.*;
import java.net.*;

public class SMTPClient {
        public static void main(String[] args) {
                if(args.length < 4) {System.out.println("Enter <server> <port> <mail from> <rcpt to>"); return;}
                try {
                        String smtpServerName = args[0];
                        int port = Integer.valueOf(args[1]).intValue();
                        Socket smptServerSocket = new Socket(args[0], port);
                        BufferedReader brSmptServerSocket = new BufferedReader(new InputStreamReader(smptServerSocket.getInputStream()));
                        PrintWriter pw = new PrintWriter(smptServerSocket.getOutputStream());
                        BufferedReader keyboard = new BufferedReader(new InputStreamReader(System.in));

                        // helo server
                        pw.println("helo " + smtpServerName);
                        pw.flush();

                        // nhan du lieu tu smpt server gui ve
                        String dataRecive = brSmptServerSocket.readLine();
                        System.out.println("Server Response : " + dataRecive);

                        // mail from
                        pw.println("mail from:" + args[2]);
                        pw.flush();
                        dataRecive = brSmptServerSocket.readLine();
                        System.out.println("Server Response: " + dataRecive);

                        // recipent to mail
                        pw.println("rcpt to:" + args[3]);
                        pw.flush();
                        dataRecive = brSmptServerSocket.readLine();
                        System.out.println("Server Response: " + dataRecive);
                        pw.println("data");
                        pw.flush();
                        dataRecive = brSmptServerSocket.readLine();
                        System.out.println("Server Response: " + dataRecive);

                        // subject
                        System.out.print("subject:");
                        String subjectMail = keyboard.readLine();
                        pw.println("subject:" + subjectMail);
                        pw.flush();
                        dataRecive = brSmptServerSocket.readLine();
```

```
            System.out.println("Server Response: " + dataRecive);

            // nhap noi dung mail
            String mailBody;
            while(true) {
                mailBody = keyboard.readLine();
                if(mailBody.equals(new String("."))) {
                    pw.println(".");
                    pw.flush();
                    break;
                }
                pw.println(mailBody);
            }
            dataRecive = brSmptServerSocket.readLine();
            System.out.println("Server Response : " + dataRecive);
            smptServerSocket.close();
        } catch(IOException e) {
            System.out.println(e);
        }
    }
}
```

POPClient.java

```
import java.io.*;
import java.util.*;
import java.net.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class POPClient {
    public static void main(String[] args) {
        if(args.length < 4) {System.out.println("Enter <server> <port> <user>
        <password>"); return;}
        try {
            String pop3ServerName = args[0];
            int port = Integer.valueOf(args[1]).intValue();

            // create socket
            Socket pop3ServerSocket = new Socket(pop3ServerName, port);
            BufferedReader brPop3ServerSocket = new BufferedReader(new
            InputStreamReader(pop3ServerSocket.getInputStream()));
            PrintWriter pw = new
            PrintWriter(pop3ServerSocket.getOutputStream());
            BufferedReader keyboard = new BufferedReader(new
            InputStreamReader(System.in));
            String dataRecive = brPop3ServerSocket.readLine();
            System.out.println("Server Response : " + dataRecive);

            // Login
            pw.println("user " + args[2]);
            pw.flush();
```

```
dataRecive = brPop3ServerSocket.readLine();
System.out.println("Server Response : " + dataRecive);
pw.println("pass " + args[3]);
pw.flush();
dataRecive = brPop3ServerSocket.readLine();
System.out.println("Server Response: " + dataRecive);

// recive a email with number
String noMail = "1";
while(true) {
    System.out.println("Enter message no: <number>, '0' for the
    end, 'inbox' for the last email: ");
    noMail = keyboard.readLine();
    if(noMail.equals("0")) {
        pw.println("QUIT");
        pw.flush();
        break;
    }

    if(noMail.equals("inbox")) {
        pw.println("stat");
        pw.flush();
        dataRecive = brPop3ServerSocket.readLine();
        System.out.println(dataRecive);
        Pattern p = Pattern.compile("\\d+");
        Matcher numberOfMail = p.matcher(dataRecive);
        if(numberOfMail.find()) {
            pw.println("retr " + numberOfMail.group());
            pw.flush();
        }
    } else {
        pw.println("retr " + noMail);
        pw.flush();
    }

    // read email
    while(true) {
    dataRecive = brPop3ServerSocket.readLine();
    if(dataRecive.equals("-ERR There's no message " + noMail +
    ".")) {
        System.out.println("-ERR There's no message " + noMail
        + ".");
        break;
    }
    System.out.println(dataRecive);
    if(dataRecive.equals(".")) {
        break;
    }
    }
}
dataRecive = brPop3ServerSocket.readLine();
```

```
            System.out.println("Server Response : " + dataRecive);
            pop3ServerSocket.close();
        } catch(IOException e) {
            System.out.println(e);
        }
    }
}

// Session 6
// PT1_Client.java
import java.rmi.*;
import java.net.MalformedURLException;
import java.util.Scanner;

public class PT1_Client {
    static void input(float[][] a, int m, int n) {
        Scanner sc = new Scanner(System.in);
        for(int i = 0; i < m; i++) {
            for(int j = 0; j < n; j++) {
                a[i][j] = sc.nextFloat();
            }
            System.out.println();
        }
    }
    static void output(float[][] c) {
        for(int i = 0; i < c.length; i++) {
            for(int j = 0; j < c[i].length; j++) {
                System.out.print(c[i][j] + " ");
            }
            System.out.println();
        }
    }
    public static void main(String[] args) {
        try {
            // Do tim doi tuong
            PT1_Itf ref = (PT1_Itf)Naming.lookup("rmi://" + args[0] +
            "/PT1Object");

            // Goi ham tren doi tuong
            float a, b;
            int nA = 0;
            Scanner sc = new Scanner(System.in);
            while(true) {
                int option;
                System.out.print("Option 1 for giaPT1, 2 for KyVong, 3 for
                multiply matrix and 0 for the end: ");
                option = sc.nextInt();
                if(option == 0) return;
                else if(option == 1) {
                    System.out.print("Nhap vao a: ");
                    a = sc.nextFloat();
                    System.out.print("Nhap vao b: ");
                    b = sc.nextFloat();
                    String result = ref.GiaiPT1(a, b);
                    System.out.println("ket qua: " + result);
                } else if(option == 2) {
                    System.out.print("Nhap vao so pt day so a: ");
                    nA = sc.nextInt();
                    float[] dayA = new float[nA];
                    for(int i = 0; i < nA; i++) {
                        dayA[i] = sc.nextFloat();
                    }
                    String result1 = ref.KyVong(dayA);
                    System.out.println("ket qua: " + result1);
                } else if(option == 3) {
                    int m, n;
                    System.out.println("Nhap vao matran A: ");
                    System.out.println("Nhap vao so hang (m) cua matran A:
                    ");
                    m = sc.nextInt();
                    System.out.println("Nhap vao so cot (n) cua matran A:
                    ");
                    n = sc.nextInt();
                    System.out.println(">>>");
                    float[][] aMatrix = new float[m][n];
                    input(aMatrix, m, n);
                    System.out.println("Nhap vao matran B: ");
                    System.out.println("Nhap vao so hang (m) cua matran B:
                    ");
                    m = sc.nextInt();
                    System.out.println("Nhap vao so cot (n) cua matran B:
                    ");
                    n = sc.nextInt();
                    System.out.println(">>>");
                    float[][] bMatrix = new float[m][n];
                    input(bMatrix, m, n);
                    float[][] cMatrix = ref.NhanMatran(aMatrix, bMatrix);
                    System.out.println("ket qua: ");
                    output(cMatrix);
                }
            }
        } catch(NotBoundException e) {
        System.out.println("Khong tim thay doi tuong");
        } catch(MalformedURLException e) {
        System.out.println("Sai trong dinh dang URL");
        } catch(RemoteException e) {
        System.out.println("Loi trong khi goi ham tu xassss");
        }
    }
}

// PT1_Itf.java
```

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface PT1_Itf extends Remote {
        public String GiaiPT1(float a, float b) throws RemoteException;
        public String KyVong(float[] a) throws RemoteException;
        public float[][] NhanMatran(float[][] a, float[][] b) throws RemoteException;
}

// PT1_Server.java
import java.rmi.*;
import java.net.MalformedURLException;

public class PT1_Server {
        public static void main(String[] args) {
                if(System.getSecurityManager() == null) // cai dat co che bao mat
                        System.setSecurityManager(new RMISecurityManager());
                try {
                        // tao doi tuong cho phep goi ham tu xa
                        PT1 obj = new PT1();
                        System.out.println("Tao object cho phep goi tu xa");
                        // Dang ky doi tuong
                        Naming.rebind("PT1Object", obj);
                        System.out.println("Dang ky thanh cong doi tuong");
                } catch(RemoteException e) {
                        System.out.println("Loi trong qua trinh tao doi tuong");
                } catch(MalformedURLException e) {
                        System.out.println("Loi khi dang ky doi tuong");
                }
        }
}

//PT1.java
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

public class PT1 extends UnicastRemoteObject implements PT1_Itf {
        public PT1() throws RemoteException {
                super();
        }

        // cai dat ham goi tu xa
        public String GiaiPT1(float a, float b) {
                if(b == 0 && (a == 0 || a != 0)) return "x = 0";
                else if(a == 0 && b != 0) return "Pt vo nghiem";
                else return "x = " + Float.toString(-b/a);
        }
        public String KyVong(float[] a) {
                float sum = 0;
                float Ea = 0;
                float Aa = 0;
                for(int i = 0; i < a.length; i++) {
```

```
                sum += a[i];
            }
            Ea = sum/a.length;
            for(int i = 0; i < a.length; i++) {
                sum += Math.pow((a[1] - Ea), 2);
            }
            Aa = sum;
            return "E = " + Float.toString(Ea) + ", " + "A = " + Float.toString(Aa);
        }
        public float[][] NhanMatran(float[][] a, float[][] b) {
            float[][] c = new float[a.length][b[0].length];
            for(int i = 0; i < a.length; i++) {
                for(int j = 0; j < a[i].length; j++) {
                    for(int t = 0; t < b[j].length; t++) {
                        c[i][t] += a[i][j] * b[j][t];
                    }
                }
            }
            return c;
        }
    }
```