SOCKET PROGRAMMING

CLIENT

```
import java.io.*;
import java.net.*;
class Client
{
	public static void main(String args[]) throws Exception
	{
		Socket s=new Socket("localhost",50);
		BufferedReader is=new BufferedReader(new InputStreamReader
(s.getInputStream()));
		PrintStream os=new PrintStream(s.getOutputStream());
		BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
		String str;
		System.out.print("Enter data to Server: ");
		str=br.readLine();
		os.println(str);
		str=is.readLine();
		System.out.println("Reply from Server: "+str);
		s.close();
	}
}
```

SERVER

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

```
                String c;
                ServerSocket ss=new ServerSocket(50);
                Socket s=ss.accept();
                BufferedReader is=new BufferedReader(new InputStreamReader
(s.getInputStream()));
                PrintStream os=new PrintStream(s.getOutputStream());
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                c=is.readLine();
                os.println(c);
                ss.close();
                s.close();
        }
}
```

OUTPUT

SERVER

```
Z:\cs2k944> javac Server.java
Z:\cs2k944> java Server
```

CLIENT

```
Z:\cs2k944> javac Client.java
Z:\cs2k944> java Client
Enter data to Server: hello
Reply from Server: hello
```

DATAGRAM

CLIENT

```
import java.io.*;
import java.net.*;
class UDPClient
{
        public static DatagramSocket clientsocket;
        public static DatagramPacket dp;
        public static BufferedReader dis;
        public static InetAddress ia;
        public static byte buf[] = new byte[1024];
        public static int cport = 789, sport = 790;
        public static void main(String args[]) throws IOException
        {
                clientsocket = new DatagramSocket(cport);
                dp = new DatagramPacket(buf, buf.length);
                dis = new BufferedReader(new InputStreamReader(System.in));
                ia = InetAddress.getLocalHost();
                System.out.println("Client is Running... Type 'STOP' to Quit");
                while(true) {
                        System.out.print("Client:");
                        String str = new String(dis.readLine());
                        buf = str.getBytes();
                        if(str.equalsIgnoreCase("STOP")) {
                                System.out.println("Terminated...");
                                clientsocket.send(new DatagramPacket(buf,str.length(), ia,sport));
                                break;
                        }
```

```
                clientsocket.send(new DatagramPacket(buf,str.length(), ia, sport));
                clientsocket.receive(dp);
                String str2 = new String(dp.getData(), 0,dp.getLength());
                System.out.println("Server: " + str2);
            }
        }
}
```

SERVER

```
import java.io.*;
import java.net.*;
class UDPServer
{
    public static DatagramSocket serversocket;
    public static DatagramPacket dp;
    public static BufferedReader dis;
    public static InetAddress ia;
    public static byte buf[] = new byte[1024];
    public static int cport = 789,sport=790;
    public static void main(String args[]) throws IOException
    {
        serversocket = new DatagramSocket(sport);
        dp = new DatagramPacket(buf,buf.length);
        dis = new BufferedReader(new InputStreamReader(System.in));
        ia = InetAddress.getLocalHost();
        System.out.println("Server is Running...");
        while(true) {
            serversocket.receive(dp);
            String str = new String(dp.getData(), 0,dp.getLength());
```

```
                if(str.equalsIgnoreCase("STOP")) {
                        System.out.println("Terminated...");
                        break;
                }
                System.out.println("Client: " + str);
                System.out.print("Server:");
                String str1 = new String(dis.readLine());
                buf = str1.getBytes();
                serversocket.send(new DatagramPacket(buf,str1.length(), ia, cport));
        }
}}
```

OUTPUT

SERVER

```
Z:\cs2k944> javac UDPServer.java
Z:\cs2k944> java UDPServer
Server is Running...
Client: hai
Server:hello
Terminated...
```

CLIENT

```
Z:\cs2k944> javac UDPClient.java
Z:\cs2k944> java UDPClient
Client is Running... Type 'STOP' to Quit
Client:hai
Server: hello
Client:stop
Terminated...
```

TCP SOCKETS

CLIENT

```
import java.net.*;
import java.io.*;
class  TCPClient
{
        public static void main(String args[]) throws Exception
        {
                Socket s=new Socket("localhost",2000);
                BufferedReader is=new BufferedReader(new
InputStreamReader(s.getInputStream()));
                PrintStream os=new PrintStream(s.getOutputStream());
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                String str;
                System.out.println("Client is Running... Type 'STOP' to Quit");
                while(true) {
                        System.out.print("Client : ");
                        str=br.readLine();
                        os.println(str);
                        str=is.readLine();
                        System.out.print("Server : "+str+"\n");
                        if ( str.equalsIgnoreCase("BYE") ) {
                                System.out.println("Terminated...");
                                break;
                        }
                }
                 s.close();
                 is.close();
```

```
                os.close();
                br.close();
        }
}
```

SERVER

```
import java.net.*;
import java.io.*;
class  TCPServer
{
        public static void main(String args[]) throws Exception
        {
                ServerSocket ss=new ServerSocket(2000);
                Socket s=ss.accept();
                BufferedReader is=new BufferedReader(new
InputStreamReader(s.getInputStream()));
                PrintStream os=new PrintStream(s.getOutputStream());
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                String str;
                System.out.println("Server is Running...");
                while(true) {
                        str=is.readLine();
                        if (str.equalsIgnoreCase("stop")) {
                                os.println("BYE");
                                System.out.println("Terminated...");
                                break;
                         }
                        System. out.print("Client : "+str+"\n");
                        System.out.print("Server : ");
                        str=br.readLine();
```

```
                os.println(str);
            }
            ss.close();
            s.close();
            is.close();
            os.close();
            br.close();
        }
}
```

OUTPUT

SERVER

```
Z:\cs2k944> javac TCPServer.java
Z:\cs2k944> java TCPServer
Server is Running...
Client : hai
Server : hello
Terminated...
```

CLIENT

```
Z:\cs2k944> javac TCPClient.java
Z:\cs2k944> java TCPClient
Client is Running... Type 'STOP' to Quit
Client : hai
Server : hello
Client : stop
Server : BYE
Terminated...
```

SMTP

SMTPClient.java

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

class SMTPClient extends JFrame implements ActionListener {
        JLabel from_lbl,pwd_lbl,to_lbl,sub_lbl,msg_lbl;
        JTextField from_txt,to_txt,sub_txt;
        JPasswordField pwd_txt;
        JTextArea msg_txt;
        JButton send_btn,cancel_btn;
        JScrollPane sp;
        SMTPClient() {
                setTitle("SMTP CLIENT PROGRAM");
                setVisible(true);
                setSize(390,400);
                setLocation(200,200);
                setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                setLayout(null);
                from_lbl=new JLabel("FROM:");
                from_lbl.setBounds(30,20,100,20);
                pwd_lbl=new JLabel("PASSWORD:");
                pwd_lbl.setBounds(30,60,100,20);
                to_lbl=new JLabel("TO:");
                to_lbl.setBounds(30,100,100,20);
                sub_lbl=new JLabel("SUBJECT:");
                sub_lbl.setBounds(30,140,100,20);
                msg_lbl=new JLabel("MESSAGE:");
                msg_lbl.setBounds(30,180,100,20);
                from_txt=new JTextField();
```

```
        from_txt.setBounds(150,20,200,20);
        pwd_txt=new JPasswordField();
        pwd_txt.setEchoChar('*');
        pwd_txt.setBounds(150,60,200,20);
        to_txt=new JTextField();
        to_txt.setBounds(150,100,200,20);
        sub_txt=new JTextField();
        sub_txt.setBounds(150,140,200,20);
        msg_txt=new JTextArea();
        sp=new JScrollPane(msg_txt);
        sp.setBounds(30,210,320,100);
        send_btn=new JButton("SEND");
        send_btn.setBounds(90,320,80,30);
        cancel_btn=new JButton("CANCEL");
        cancel_btn.setBounds(210,320,80,30);
        add(from_lbl);
        add(from_txt);
        add(pwd_lbl);
        add(pwd_txt);
        add(to_lbl);
        add(to_txt);
        add(sub_lbl);
        add(sub_txt);
        add(msg_lbl);
        add(sp);
        add(send_btn);
        add(cancel_btn);
        send_btn.addActionListener(this);
        cancel_btn.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae) {
        Object x=ae.getSource();
        if(x==send_btn) {
```

```
                    new MyMail(from_txt.getText(), pwd_txt.getText(), to_txt.getText(),
sub_txt.getText(), msg_txt.getText()).SendMail();
                }
                if(x==cancel_btn) {
                    System.exit(0);
                }
        }
        public static void main(String args[]) {
                new SMTPClient();
        }
}
```

MyMail.java

```
import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.swing.JOptionPane;

public class MyMail {
        String host,from,pass,to,sub,msg;
        MyMail(String from,String pass,String to,String sub,String msg) {
                host = "smtp.gmail.com";
                this.from = from;
                this.pass = pass;
                this.to = to;
                this.sub = sub;
```

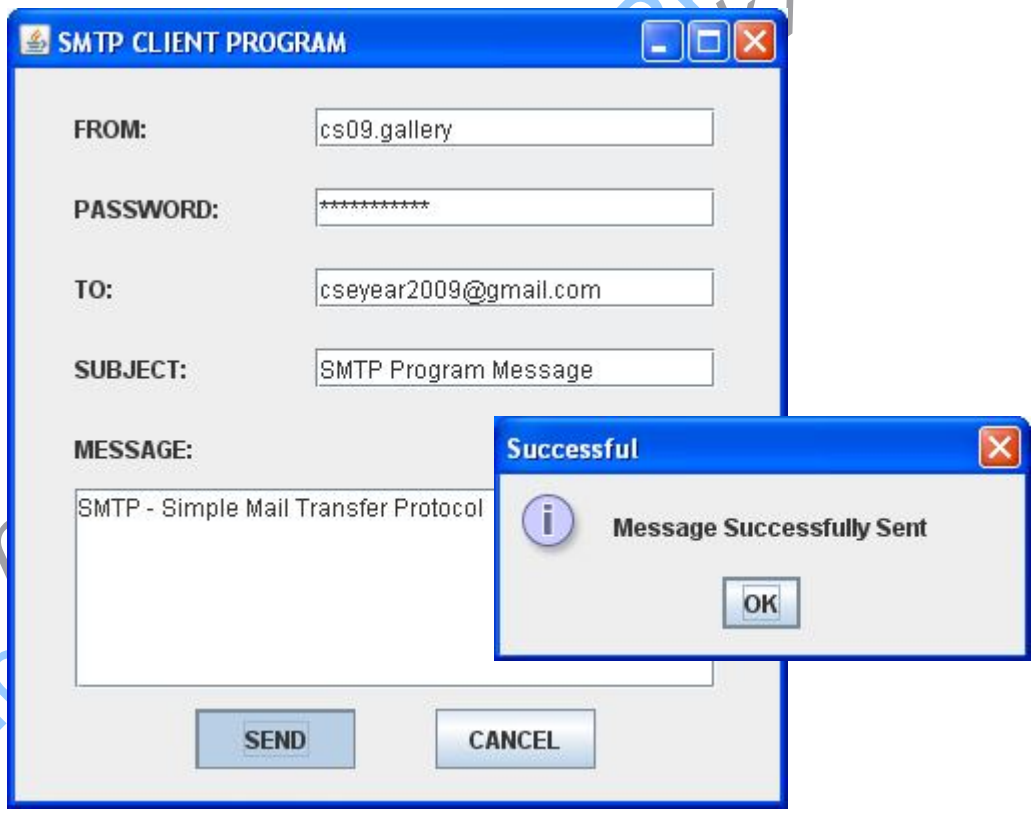
```
        this.msg = msg;
    }
    public void SendMail() {
        host = "smtp.gmail.com";
        Properties props = System.getProperties();
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", host);
        props.put("mail.smtp.user", from);
        props.put("mail.smtp.password", pass);
        props.put("mail.smtp.port", 587);
        props.put("mail.smtp.auth", "true");
        try{
        Session session = Session.getDefaultInstance(props, null);
        MimeMessage message = new MimeMessage(session);
        message.setFrom(new InternetAddress(from));
        message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
        message.setSubject(sub);
        message.setText(msg);
        Transport transport = session.getTransport("smtp");
        transport.connect(host,from, pass);
        System.out.println("connected");
        transport.sendMessage(message, message.getAllRecipients());
        System.out.println("Sent message successfully....");
        JOptionPane.showMessageDialog(null,"Message Successfully
Sent","Successful",JOptionPane.INFORMATION_MESSAGE);
        transport.close();
        System.exit(0);
        }
        catch (MessagingException me)
        {
```

```
                    JOptionPane.showMessageDialog(null,me,"Warning Message",
JOptionPane.WARNING_MESSAGE);
                }
        }
}
```

OUTPUT

Z:\cs2k944> javac SMTPClient.java

Z:\cs2k944> java SMTPClient



FILE TRANSFER USING TCP SOCKETS

CLIENT

```
import java.io.*;
import java.net.*;
class FTPClient
{
	public static void main(String srgs[])throws IOException
	{
		Socket s=null;
		BufferedReader get=null;
		PrintWriter put=null;
		try
		{
			s=new Socket("localhost",8081);
			get=new BufferedReader(new InputStreamReader(s.getInputStream()));
			put=new PrintWriter(s.getOutputStream(),true);
		}
		catch(Exception e) {
			System.exit(0);
		}
		String f;
		int u;
		System.out.println("Enter the file name to transfer from server:");
		DataInputStream dis=new DataInputStream(System.in);
		f=dis.readLine();
		put.println(f);
		File f1=new File(f);
		FileOutputStream  fs=new FileOutputStream(f1);
```

```
            while((u=get.read())!=-1) {
                fs.write((char)u);
            }
            fs.close();
            System.out.println("File received");
            s.close();
        }
}
```

SERVER

```
import java.io.*;
import java.net.*;
class FTPServer
{
    public static void main(String args[])throws IOException
    {
        ServerSocket ss=null;
        try
        {
            ss=new ServerSocket(8081);
        }
        catch(IOException e)
        {
            System.out.println("couldn't listen");
            System.exit(0);
        }
        Socket cs=null;
        try
        {
```

```
                cs=ss.accept();
                System.out.println("Connection established"+cs);
            }
            catch(Exception e)
            {
                System.out.println("Accept failed");
                System.exit(1);
            }
            PrintWriter put=new PrintWriter(cs.getOutputStream(),true);
            BufferedReader st=new BufferedReader(new
InputStreamReader(cs.getInputStream()));
            String s=st.readLine();
            System.out.println("The requested file is: "+s);
            File f=new File(s);
            if(f.exists()) {
                BufferedReader d=new BufferedReader(new FileReader(s));
                int line;
                while((line=d.read())!=-1) {
                    put.write(line);
                    put.flush();
                }
                d.close();
                System.out.println("File Transferred");
                cs.close();
                ss.close();
            }
        }
}
```

OUTPUT

SERVER

Z:\cs2k944> javac FTPServer.java

Z:\cs2k944> java FTPServer

Connection establishedSocket[addr=/127.0.0.1,port=1030,localport=8081]

The requested file is: cns.txt

File Transferred

CLIENT

Z:\cs2k944> javac FTPClient.java

Z:\cs2k944> java FTPClient

Enter the file name to transfer from server:

cns.txt

File received

IMPLEMENTATION OF ANY TWO CONGESTION CONTROL ALGORITHMS

a) LEAKY BUCKET ALGORITHM

```
import java.io.*;

class LeakyBucket extends Thread {
        static int delay=1,time=0;
        public static void bktInput(int a,int b,int size) throws Exception {
                if(a>size)
                        System.out.println("BUCKET OVERFLOW");
                else {
                        Thread.sleep(1);
                        time++;
                        while(a>b) {
                                System.out.println("Time: "+time +"seconds");
                                System.out.println(b+"bytes outputted");
                                a-=b;
                                Thread.sleep(1);
                                time++;
                        }
                        System.out.println("Time: "+time +"seconds");
                        if(a>0)
                                System.out.println("Last "+a+" bytes sent");
                        System.out.println("BUCKET OUTPUT SUCCESSFUL.....");
                }
        }
```

```
    public static void main(String args[])throws Exception {
        int bkt_size,output_rate,no_pkt,pkt_size,i;
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the Bucket Size:");
        bkt_size=Integer.parseInt(br.readLine());
        System.out.println("Enter the Output Rate:");
        output_rate=Integer.parseInt(br.readLine());
        System.out.println("Enter No. of Packets:");
        no_pkt=Integer.parseInt(br.readLine());
        for(i=1;i<=no_pkt;i++) {
            Thread.sleep(delay);
            time+=delay;
            pkt_size=(int)(Math.random()*1000);
            System.out.println("\n"+"Time: "+time+"seconds");
            System.out.println("Packet No. "+i+"  Packet Size = "+pkt_size);
            bktInput(pkt_size,output_rate,bkt_size);
            delay=(int)(Math.random()*10);
        }
    }
}
```

OUTPUT

```
Z:\cs2k944> javac LeakyBucket.java
Z:\cs2k944> java LeakyBucket
Enter the Bucket Size:
512
Enter the Output Rate:
256
Enter No. of Packets:
4
```

Time: 1seconds
Packet No. 1 Packet Size = 99
Time: 2seconds
Last 99 bytes sent
BUCKET OUTPUT SUCCESSFUL.....
Time: 10seconds
Packet No. 2 Packet Size = 831
BUCKET OVERFLOW

Time: 10seconds
Packet No. 3 Packet Size = 283
Time: 11seconds
256bytes outputted
Time: 12seconds
Last 27 bytes sent
BUCKET OUTPUT SUCCESSFUL.....

Time: 21seconds
Packet No. 4 Packet Size = 861
BUCKET OVERFLOW

b) TOKEN BUCKET ALGORITHM

```
import java.io.*;

class TokenBucket extends Thread {
	static int token=0,time=0,delay=1;
	public static void bktInput(int i,int ps,int bs) {
		if(ps>token) {
			System.out.println("Waiting for enough token");
			time++;
			System.out.println("\n"+"Time: "+time+"seconds");
			token+=10;
			System.out.println("No. of Token = "+token);
			bktInput(i,ps,bs);
		}
		else {
			token=token-ps;
			System.out.println("PACKET NO."+i+" IS OUTPUTTED
SUCCESSFULLY...");
		}
	}
	public static void main(String args[])throws Exception {
		int bkt_size,no_pkt,pkt_size,i;
		BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
		System.out.println("Enter the Bucket Size[>=50]:");
		bkt_size=Integer.parseInt(br.readLine());
		System.out.println("Enter No. of Packets:");
		no_pkt=Integer.parseInt(br.readLine());
		for(i=1;i<=no_pkt;i++) {
			Thread.sleep(delay);
```

```
                time+=delay;
                token+=(delay*10);
                System.out.println("\n"+"Time: "+time+"seconds");
                if(token>bkt_size) {
                        System.out.println("BUCKET OVERFLOW");
                        token=bkt_size;
                }
                pkt_size=(int)(Math.random()*50);
                System.out.println("No. of Token = "+token);
                System.out.println("Packet No.: "+i+"  Packet Size: "+pkt_size);
                bktInput(i,pkt_size,bkt_size);
                delay=(int)(Math.random()*10);
        }
    }
}
```

OUTPUT

```
Z:\cs2k944> javac TokenBucket.java
Z:\cs2k944> java TokenBucket
Enter the Bucket Size[>=50]:
60
Enter No. of Packets:
4

Time: 1seconds
No. of Token = 10
Packet No.: 1  Packet Size: 20
Waiting for enough token
```

Time: 2seconds

No. of Token = 20

PACKET NO.1 IS OUTPUTTED SUCCESSFULLY...

Time: 3seconds

No. of Token = 10

Packet No.: 2 Packet Size: 22

Waiting for enough token

Time: 4seconds

No. of Token = 20

Waiting for enough token

Time: 5seconds

No. of Token = 30

PACKET NO.2 IS OUTPUTTED SUCCESSFULLY...

Time: 8seconds

No. of Token = 38

Packet No.: 3 Packet Size: 1

PACKET NO.3 IS OUTPUTTED SUCCESSFULLY...

Time: 16seconds

BUCKET OVERFLOW

No. of Token = 60

Packet No.: 4 Packet Size: 15

PACKET NO.4 IS OUTPUTTED SUCCESSFULLY...