userApplication.java

```
//XRYSOYLA TSIMPERI AEM:9272
package userApplication;

import java.net.*;
import java.util.Scanner;
import java.io.*;
import javax.sound.sampled.*;


public class userApplication {

	private static String  clientPort 		= "48024";
	private static String   serverPort 		= "38024";
	private static String echoCode 			= "E3907";
	private static String imageCode 		= "M4195";
	private static String soundCodeDPCM 		= "A4224F999";
	private static String soundCodeAQDPCM 		="A4224AQF999";
	private static String ithakiCopterCode 		= "Q4036";
	private static String obdVehicleCode 		= "V5922";
	private static byte[] hostIP 			= { (byte) 155, (byte) 207, 18,
(byte) 208 };
	private static InetAddress hostAddress;
	private static Scanner input;

	public static void main(String[] args) throws IOException, LineUnavailableException {
		hostAddress = InetAddress.getByAddress( hostIP );

		System.out.println("Choose one option :\n"
				+ "1. Response times from echo packets.\n"
				+ "2. Temperature packets.\n"
				+ "3. Camera image.\n"
				+ "4. Frequency Generator Packet Transmission.\n"
				+ "5. DPCM sound.\n"
				+ "6. AQDPCM sound.\n"
				+ "7. IthakiCopter telemetry.\n"
				+ "8. Onboard diagnostics for the vehicle.\n"
				+ "9. All\n"
				+ "0. Exit\n");

		input = new Scanner(System.in);
		int choice = input.nextInt();

		switch (choice) {
		case 1: {

			System.out.println("echo packets with delay press 1, with no added
delay press 0\n");
			input = new Scanner(System.in);
			int c = input.nextInt();
			if(c==1){
			    echoPacket();
			    break;
			}
			else{
				echoCode="E0000";
			    echoPacket();
			break;
			}
			//break;
		}
		case 2: {
			 echoCode = echoCode + "T";
			 echoPacketTemperature();
			break;
		}
		case 3: {

			System.out.println("For CAM=FIX  press 1,for CAM=PTZ 0\n");
			input = new Scanner(System.in);
			int c = input.nextInt();
			if(c==1){
```

```
                        imageReceive();
                        break;
                    }
                    else{
                            imageCode=imageCode+"PTZ";
                            imageReceive();
                            break;

                    }



                }
                case 4: {

                        soundDPCM();
                        break;
                }

                case 5: {
                        soundDPCM();
                        break;
                }
                case 6: {
                        soundAQDPCM();
                        break;
                }
                case 7: {
                        ithakiCopter();
                        break;
                }
                case 8:{
                        VehicleDiagnostics();
                        break;
                }
                case 9: {

                    echoPacket();
                    echoPacketTemperature();
                        imageReceive();
                        soundDPCM();
                        soundAQDPCM();
                        ithakiCopter();
                        VehicleDiagnostics();
                        break;
                }
                case 0: {
                        System.out.println("Exit\n");
                        return;
                }

                default: System.out.println("Please choose an option from 0 to 9");

                }

        }

//-----------------------------Receive Echo Packets----------------------------
        public static void echoPacket() throws IOException {

                System.out.println("Receiving Echo packets .....\n");
                int serverport, clientport;
                int cnt = 0;
                long endTime;
                long sendTime = 0;
                long receiveTime = 0;
                byte[] txbuffer, rxbuffer;
                String packetInfo;
                String message = "";
                String responseTime = "";

                PrintWriter resTime = new PrintWriter("responseTime.txt");
                PrintWriter msgs = new PrintWriter("msgs.txt");
```

```
                DatagramSocket s = new DatagramSocket();

                packetInfo = echoCode;
                txbuffer = packetInfo.getBytes();
                serverport = Integer.parseInt(serverPort);
                byte[] hostIP = { (byte)155 , (byte)207 , 18 , (byte)208 };
                InetAddress hostAddress = InetAddress.getByAddress(hostIP);

                DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, hostAddress,
serverport);

                clientport = Integer.parseInt(clientPort);
                DatagramSocket r = new DatagramSocket(clientport);

                r.setSoTimeout(5000);
                rxbuffer = new byte[32];

                DatagramPacket q = new DatagramPacket(rxbuffer, rxbuffer.length);

                endTime = System.currentTimeMillis() + 240000;

                while(System.currentTimeMillis() <= endTime) {
                        sendTime = System.currentTimeMillis();
                        s.send(p);
                        cnt++;

                        try {
                                r.receive(q);
                                receiveTime = System.currentTimeMillis();
                                message = new String(rxbuffer, 0, q.getLength());
                        } catch (Exception x) {
                                System.out.println(x);
                        }

                        responseTime = String.valueOf(receiveTime - sendTime);
                        resTime.println(responseTime);
                        msgs.println(message);
                }

                System.out.printf("\n Finished....Number of echo packets received: %d", cnt);

                s.close();
                r.close();
                resTime.close();
                msgs.close();

        }

//-----------------------------Receive Temperature Packets----------------------------
public static void echoPacketTemperature() throws IOException {

                int serverport, clientport, i;
                int cnt = 0;
                byte[] txbuffer, rxbuffer;
                String packetInfo;
                String message = "";
                String echoTemperatureCode = "";

                PrintWriter tempMsgs = new PrintWriter("tempMsgs.txt");

                DatagramSocket s = new DatagramSocket();

                packetInfo = echoCode + "T";
                txbuffer = packetInfo.getBytes();
                serverport = Integer.parseInt(serverPort);
                byte[] hostIP = { (byte)155 , (byte)207 , 18 , (byte)208 };
                InetAddress hostAddress = InetAddress.getByAddress(hostIP);

                DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, hostAddress,
serverport);

                clientport = Integer.parseInt(clientPort);
                DatagramSocket r = new DatagramSocket(clientport);
```

```
                r.setSoTimeout(8000);
                rxbuffer = new byte[54];

                DatagramPacket q = new DatagramPacket(rxbuffer, rxbuffer.length);

                for (i=0; i<100; i++) {

                        if(i<10)
                                echoTemperatureCode = echoCode + "T0" + String.valueOf(i);
                        else
                                echoTemperatureCode = echoCode + "T" + String.valueOf(i);

                        txbuffer = echoTemperatureCode.getBytes();

                        p.setData(txbuffer);
                        p.setLength(txbuffer.length);

                        s.send(p);
                        cnt++;

                        try {
                                r.receive(q);
                                message = new String(rxbuffer, 0, q.getLength());
                        } catch (Exception x) {
                                System.out.println(x);
                        }

                        tempMsgs.println(message);
                }

                System.out.printf("\n Finished.....Number of temperature packets received :
%d", cnt);

                s.close();
                r.close();
                tempMsgs.close();

        }

//-----------------------------Receive Image-----------------------------
        public static void imageReceive() throws IOException {

                int serverport, clientport;
                byte[] txbuffer, rxbuffer;

                FileOutputStream image = new FileOutputStream ("image.jpg");

                DatagramSocket s = new DatagramSocket();

                txbuffer = imageCode.getBytes();
                serverport = Integer.parseInt(serverPort);
                byte[] hostIP = { (byte)155, (byte)207, 18, (byte)208 };
                InetAddress hostAddress = InetAddress.getByAddress(hostIP);

                DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, hostAddress,
serverport);

                clientport = Integer.parseInt(clientPort);
                try (DatagramSocket r = new DatagramSocket(clientport)) {
                        r.setSoTimeout(9000);
                        rxbuffer = new byte[128];

                        DatagramPacket q = new DatagramPacket(rxbuffer, rxbuffer.length);

                        s.send(p);

                        for(;;) {
                                try {
                                        r.receive(q);
                                        image.write(rxbuffer);
                                        image.flush();
                                } catch(Exception x) {
                                        System.out.println(x);
                                        break;
```

```
                        }
                }

                System.out.println("Finished....image received\n");
                r.close();
        }

        s.close();
        image.close();

    }

//-----------------------------Receive DPCM Sound Clip----------------------------
    public static void soundDPCM() throws IOException, LineUnavailableException {

        int serverport, clientport;
        int i = 0;
        int bpp = 128;
        int soundPackets = 999;
        int samplesSize = 2*bpp*soundPackets;
        int[] soundSamples = new int[samplesSize];
        int[] soundClip = new int[samplesSize];
        byte[] txbuffer, rxbuffer;
        byte[] receivedSound = new byte[bpp*soundPackets];

        PrintWriter samples = new PrintWriter("SamplesDPCM.txt");
        PrintWriter samplesDiffs = new PrintWriter("SamplesDiffsDPCM.txt");


        DatagramSocket s = new DatagramSocket();

        txbuffer = soundCodeDPCM.getBytes();
        serverport = Integer.parseInt(serverPort);
        byte[] hostIP = { (byte)155, (byte)207, 18, (byte)208 };
        InetAddress hostAddress = InetAddress.getByAddress(hostIP);

        DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, hostAddress,
serverport);

        clientport = Integer.parseInt(clientPort);
        try (DatagramSocket r = new DatagramSocket(clientport)) {
                r.setSoTimeout(7000);
                rxbuffer = new byte[bpp];

                DatagramPacket q = new DatagramPacket(rxbuffer, rxbuffer.length);

                s.send(p);

                for(;;) {
                        try {
                                r.receive(q);
                                for( int j=0; j<bpp; j++)
                                        receivedSound[i*bpp + j] = rxbuffer[j];
                                i++;
                        } catch (Exception x) {
                                System.out.println(x);
                                break;
                        }
                }

                System.out.printf("\nNumber of DPCM packets received: %d", i);

                AudioFormat linearPCM = new AudioFormat(8000, 8, 1, true, false);
                SourceDataLine lineOut = AudioSystem.getSourceDataLine(linearPCM);
                lineOut.open(linearPCM, samplesSize);

                for (i=0; i<bpp*soundPackets; i++) {
                        int k = (int)receivedSound[i];
                        soundSamples[2*i] = (((k >> 4) & 15) - 8);
                        soundSamples[2*i + 1] = ((k & 15) - 8);
                }


                byte[] audioBufferOut = new byte[2*bpp*soundPackets];
```

```
                        for (i=1; i<samplesSize; i++)
                                soundClip[i] = soundSamples[i];

                        audioBufferOut[0] = (byte)(2*soundClip[0]);
                        for(i=1; i<samplesSize; i++) {
                                soundClip[i] =  2*soundClip[i] + audioBufferOut[i-1];
                                audioBufferOut[i] = (byte)soundClip[i];
                        }

                        for(i=0; i<samplesSize; i++){
                                samples.println(audioBufferOut[i]);
                                samplesDiffs.println(soundSamples[i]);
                        }

                        lineOut.start();
                        lineOut.write(audioBufferOut, 0, samplesSize);
                        lineOut.stop();

                        lineOut.close();
                        s.close();
                        r.close();
                }

                samples.close();
                samplesDiffs.close();

        }
//-----------------------------Receive AQ-DPCM Sound Clip-----------------------------
        public static void soundAQDPCM() throws IOException, LineUnavailableException {

                int serverport, clientport;
                int i = 0, cnt = 0;
                int bpp = 132;
                int soundPackets = 999;
                int samplesSize = 2*bpp*soundPackets;
                int lsb = 0;
                int msb = 0;
                int[] mean = new int[soundPackets];
                int[] step = new int[soundPackets];
                int[] soundSamples = new int[samplesSize];
                int[] soundClip = new int[samplesSize];
                byte[] txbuffer, rxbuffer;
                byte[] receivedSound = new byte[bpp*soundPackets];

                PrintWriter samples = new PrintWriter("SamplesAQDPCM.txt");
                PrintWriter samplesDiffs = new PrintWriter("SamplesDiffsAQDPCM.txt");
                PrintWriter means = new PrintWriter("means.txt");
                PrintWriter steps = new PrintWriter("steps.txt");

                DatagramSocket s = new DatagramSocket();

                txbuffer = soundCodeAQDPCM.getBytes();
                serverport = Integer.parseInt(serverPort);
                byte[] hostIP = { (byte)155, (byte)207, 18, (byte)208 };
                InetAddress hostAddress = InetAddress.getByAddress(hostIP);

                DatagramPacket p = new DatagramPacket(txbuffer, txbuffer.length, hostAddress,
serverport);

                clientport = Integer.parseInt(clientPort);
                try (DatagramSocket r = new DatagramSocket(clientport)) {
                        r.setSoTimeout(1000);
                        rxbuffer = new byte[bpp];

                        DatagramPacket q = new DatagramPacket(rxbuffer, rxbuffer.length);

                        s.send(p);

                        for(;;) {
                                try {
                                        r.receive(q);
                                        for(int j=0; j<bpp; j++)
                                                receivedSound[i*bpp + j] = rxbuffer[j];
```

```
                                        i++;
                                } catch (Exception x) {
                                        System.out.println(x);
                                        break;
                                }
                        }

                        System.out.printf("\nNumber of AQ-DPCM sound packets received: %d",
i);

                        AudioFormat linearPCM = new AudioFormat(8000, 16, 1, true, false);
                        SourceDataLine lineOut = AudioSystem.getSourceDataLine(linearPCM);
                        lineOut.open(linearPCM, samplesSize);

                        for (i=0; i<soundPackets; i++) {
                                lsb = (int)receivedSound[bpp*i];
                                msb = (int)receivedSound[bpp*i + 1];
                                mean[i] = (256 * msb) + (lsb & 0x00FF);
                                lsb = (int)receivedSound[bpp*i + 2];
                                msb = (int)receivedSound[bpp*i + 3];
                                step[i] = (256 * (msb & 0x00FF)) + (lsb & 0x00FF);
                        }

                        for(i=0; i<soundPackets; i++) {
                                means.println(mean[i]);
                                steps.println(step[i]);
                        }

                        for (i=0; i<soundPackets; i++) {
                                for(int j=4; j<bpp; j++) {
                                        int k = (int)receivedSound[i*bpp + j];
                                        soundSamples[2*cnt] = (((k >> 4) & 15) - 8)*step[i];
                                        soundSamples[2*cnt + 1] = ((k & 15) - 8)*step[i];
                                        cnt++;
                                }
                        }

                        byte[] audioBufferOut = new byte[512*soundPackets]; //512 =
128*2*soundPackets*2

                        for(i=1; i<256*soundPackets; i++)
                                soundClip[i] = soundSamples[i];

                        for(i=0; i<soundPackets; i++) {
                                for(int j=0; j<256; j++) {
                                        if(i==0 && j==0) continue;
                                                soundClip[i*256 + j] =  soundClip[i*256 + j]
+ soundClip[i*256 + j - 1];
                                }
                        }

                        for(i=0; i<256*soundPackets; i++){
                                audioBufferOut[2*i] = (byte)(soundClip[i] & 0xFF);
                                audioBufferOut[2*i + 1] = (byte)((soundClip[i] >> 8) & 0xFF);
                        }

                        for(i=0; i<256*soundPackets; i++) {
                                samples.println(audioBufferOut[i]);
                                samplesDiffs.println(soundSamples[i]);
                        }

                        lineOut.start();
                        lineOut.write(audioBufferOut, 0, audioBufferOut.length);
                        lineOut.stop();

                        lineOut.close();
                        s.close();
                        r.close();
                }

                samples.close();
                samplesDiffs.close();
                means.close();
                steps.close();
```

```
        }
//-----------------------------Receive Ithaki Copter Telemetry----------------------------
public static void ithakiCopter() throws IOException {

                int portNumber = 38048;
                byte[] addressArray = { (byte)155, (byte)207, 18, (byte)208 };
                InetAddress hostAddress = InetAddress.getByAddress(addressArray);

                PrintWriter copter = new PrintWriter("copter.txt");

                try (Socket s = new Socket(hostAddress, portNumber)) {
                        s.setSoTimeout(2000);
                        InputStream in = s.getInputStream();
                        OutputStream out = s.getOutputStream();

                        for(;;) {
                                try {
                                        out.write("AUTO FLIGHTLEVEL=150 LMOTOR=125 RMOTOR=125
PILOT \r\n".getBytes());
                                        int k = in.read();
                                        System.out.print((char)k);
                                        copter.print((char) k);
                                } catch(Exception x) {
                                        System.out.println(x);
                                        break;
                                }
                        }

                        s.close();
                }
                copter.close();

        }



 //-------------Receive OnBoard Car Fault Diagnostics (OBD-II) Packets Function---------------
------


        public static void VehicleDiagnostics() throws IOException {

        long endTime;
                int portNumber = 29078;

                byte[] addressArray = { (byte)155, (byte)207, 18, (byte)208 };
                InetAddress hostAddress = InetAddress.getByAddress(addressArray);


                PrintWriter diagnostic;
                String obdcode= "01 11\r";

                Socket send = new Socket(hostAddress, portNumber);
                send.setSoTimeout(1000);
                InputStream in = send.getInputStream();
                OutputStream out = send.getOutputStream();


        System.out.println("Choose one of the following options:\n"
                +"1. Engine Run Time\n"
                +"2. Intake air temperataure\n"
                +"3. Throttle position\n"
                +"4. Engine RPM\n"
                +"5. Vehicle speed\n"
                +"6. Coolant temperature\n");


        input = new Scanner(System.in);
                int choice = input.nextInt();
```

```
        switch(choice){
                case 1:{
                        diagnostic = new PrintWriter("EngineRunTime.txt");
                        obdcode= "01 1F\r";
                        break;
        }
                case 2:{
                         diagnostic = new PrintWriter("IntakeAirPressure.txt");
                         obdcode= "01 0F\r";
                         break;
        }
                case 3:{
                         diagnostic = new PrintWriter("ThrottlePosition.txt");
                         obdcode= "01 11\r";
                         break;
        }
                case 4:{
                         diagnostic = new PrintWriter("RPM.txt");
                         obdcode= "01 0C\r";
                         break;
        }
                case 5:{
                         diagnostic = new PrintWriter("Speed.txt");
                         obdcode= "01 0D\r";
                         break;
        }
                case 6:{
                         diagnostic = new PrintWriter("CarTemperature.txt");
                          obdcode= "01 05\r";
                          break;
        }
                default:{
                        diagnostic = new PrintWriter("CarTemperature.txt");
                          obdcode= "01 05\r";
                                        break    ;

                }

        }

        endTime = System.currentTimeMillis() + 240000;


        while(System.currentTimeMillis() <= endTime) {
                        try {
                                out.write(obdcode.getBytes());
                                int k = in.read();
                                System.out.print((char)k);
                                diagnostic.print((char)k);

                        } catch(Exception x) {
                                System.out.println(x);
                                break;
                        }
                }

                send.close();
                diagnostic.close();
                System.out.println("Diagnostics finished");
        }


}
```

//////////////////////////**END**////////////////////////////////////

Network.java

```
//XRYSOYLA TSIMPERI AEM:9272
import java.io.*;
import java.util.Arrays;
import java.util.Scanner;
import java.lang.String;
import java.net.*;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.SourceDataLine;

public class Network {
   DatagramSocket send;
   DatagramSocket receive;
   DatagramPacket p;
   DatagramPacket q;
   DatagramPacket f;
   DatagramPacket img;
   byte[] rxbuffer;
   int clientPort;
   int serverPort;
   String packetInfoecho;
   String packetInfoimage;
   String packetInfoaudio;

 //  String packetInfoTemp;
   byte[] txbuffer;
   byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 };
   InetAddress hostAddress;

        public Network(int clientport,int serverport) throws SocketException,
UnknownHostException {
        //Constructor initializing Network connection
        clientPort=clientport;
        serverPort=serverport;
        receive = new DatagramSocket(clientPort);
        send = new DatagramSocket();
        hostAddress = InetAddress.getByAddress(hostIP);


        receive.setSoTimeout(4000);
        rxbuffer = new byte[2048];
        q = new DatagramPacket(rxbuffer,rxbuffer.length);



        }
//-----------------------------------------------------------------------
        public static void main(String[] args) throws java.lang.Exception{
                String s=null;
                int min;
                int camera;
                int packages;
                BufferedReader readconsole = new BufferedReader(new
InputStreamReader(System.in));
                int clientport, serverport;
                int answer;

try{

            System.out.println("Type client port");
            s=readconsole.readLine();
           clientport=Integer.parseInt(s);

            System.out.println("Type server port");
            s=readconsole.readLine();
            serverport=Integer.parseInt(s);

            Network nC = new Network(clientport,serverport);
```

```
  //while (true){
                System.out.println("Press: '1' for echo request / '2' for Echo without deleay
request / '3' for temperatures request / '4' for image request / '5' for Music request / '6'
to exit ");
                s = readconsole.readLine();
                answer=Integer.parseInt( s );
                switch( answer ){
                case 1:{
                System.out.println("Type Echo request code");
                    s=readconsole.readLine();
                    nC.packetInfoecho=s;
                    System.out.println("Give runtime (minutes)");
                    s=readconsole.readLine();
                    min=Integer.parseInt(s);
                nC.ECHO(min);
                break;}
                case 2:{
                System.out.println("Type Echo request code");
                    s=readconsole.readLine();
                    nC.packetInfoecho=s;
                    System.out.println("Give runtime (minutes)");
                    s=readconsole.readLine();
                    min=Integer.parseInt(s);
                nC.ECHO_noDelay(min);
                break;}
                case 3:{
                System.out.println("Type Echo request code");
                    s=readconsole.readLine();
                    nC.packetInfoecho=s;
                nC.getTemp();
                break;}
                case 4:{
                System.out.println("Type Image request code");
                    s=readconsole.readLine();
                    nC.packetInfoimage=s;
                System.out.println("Select camera:");
                s=readconsole.readLine();
                    camera=Integer.parseInt(s);
                nC.image(camera);
                break;}
                case 5:{
                System.out.println("Type Sound request code");
                    s=readconsole.readLine();
                    nC.packetInfoaudio=s;
                System.out.println("Number of Packages? (100<=packages<=999:");
                s=readconsole.readLine();
                packages=Integer.parseInt(s);
                System.out.println("Select Modulation. Type 'DPCM' or 'AQDPCM'");
                s=readconsole.readLine();
                nC.Audio(packages, s);
                break;}
                case 6:{ System.exit(0);}
                answer=0;
                default:
                System.out.println( "Try again" );
               }
                System.out.println();
                System.out.println("KALA XRISTOUGENNA!!!! =]");
}
   catch (Exception x) {
        System.out.println(x);
        }

          }

//---------------------------------------------------------------------
        public void ECHO(int min){

                txbuffer = packetInfoecho.getBytes();
                p = new DatagramPacket(txbuffer,txbuffer.length, hostAddress,serverPort);

                long start,i,timereceived,session=32000;
                int numofpack=0,sessioncount=1,secs=1,sum=0,timeslots=32;
```

```
        int [] packpersec=new int [4000];
        int  throughput= 0;
        long runtime=min*60*1000;
        long []trans = new long[ 400000 ];
        int thr=0;
        int count=0;
        PrintStream trp = null;
        PrintStream transmission = null;
        PrintStream timepassed = null;

        try {
                  FileWriter echofile = new FileWriter("echofile.txt");
          BufferedWriter output = new BufferedWriter(echofile);

          trp= new PrintStream(new FileOutputStream("throughput.txt"));
          transmission = new PrintStream(new FileOutputStream("transmission.txt"));
          timepassed = new PrintStream(new FileOutputStream("time.txt"));

        send.send(p);

        start=System.currentTimeMillis();
        i=start;
        while (i-start<=runtime) {
                try{
                receive.receive(q);
                timereceived=System.currentTimeMillis();
                numofpack=numofpack+1;

                trans[count++]=timereceived-i;
                String message = new String(rxbuffer,0,q.getLength());
                System.out.println(message);
                output.write(message,18,8);
      output.newLine();
      transmission.println(timereceived-i);
      timepassed.println((timereceived-start));

     //calculations for throughput are not the best because i had no time to work with
Threads
      if (timereceived-start>=secs*1000){
          if (timereceived-start<(secs+1)*1000/2){
              packpersec[secs]=numofpack;
              System.out.println(numofpack);
          }
          else{
                  packpersec[secs]=0;
              secs+=1;
              packpersec[secs]=numofpack;}
         // calculate throughput
         if ( secs>=timeslots){

              for (int j=1;j<=timeslots;j++){
                 sum=sum+packpersec[secs-timeslots+j];
              }
         //       System.out.println(sum);
              thr= sum*q.getLength()*8/timeslots;
             System.out.println(thr);
             trp.println(thr);

            // sessioncount+=1;
              sum=0;
          }

     //    System.out.println(q.getLength());
          secs+=1;
          numofpack=0;
      }
                  }
      catch( Exception e ){
          System.out.println("Failed to fetch data package after 4sec");
      }
      send.send(p);
      i=System.currentTimeMillis();
    }
```

```
                output.close();

        }
        catch (Exception x) {
                        System.out.println(x);

                        }

    }
//-----------------------------------------------------------------
public void ECHO_noDelay(int min){

            String packetInfo1;
                long start,i,trans,timereceived,session=32000;
                int numofpack=0,sessioncount=1,secs=1,sum=0,timeslots=32;
                int [] packpersec=new int [4000];
                int [] throughput= new int [4000];
                long runtime=min*60*1000;

                PrintStream trp = null;
                PrintStream transmission = null;
                PrintStream timepassed = null;

                try {
                          FileWriter echofile = new FileWriter("echofile2.txt");
                  BufferedWriter output = new BufferedWriter(echofile);

                  trp= new PrintStream(new FileOutputStream("thoughput2.txt"));
                  transmission = new PrintStream(new FileOutputStream("transmission2.txt"));
                  timepassed = new PrintStream(new FileOutputStream("time2.txt"));

                  txbuffer = packetInfoecho.getBytes();
                  p = new DatagramPacket(txbuffer,txbuffer.length, hostAddress,serverPort);

                send.send(p);

                start=System.currentTimeMillis();
                i=start;
                packetInfo1 = "E0000";
                f = new DatagramPacket(packetInfo1.getBytes(),packetInfo1.getBytes().length,
hostAddress,serverPort);

                while (i-start<=runtime) {
                try{
                        receive.receive(q);

                        timereceived=System.currentTimeMillis();
                        numofpack=numofpack+1;

                        trans=timereceived-i;
                        String message = new String(rxbuffer,0,q.getLength());

                        output.write(message,18,8);
             output.newLine();
             transmission.println(trans);
             System.out.println(trans);
             timepassed.println((timereceived-start));

             // save the number of packages received per sec
             if (timereceived-start>secs*1000){
                 packpersec[secs]=numofpack;
            //   System.out.println(numofpack);

                // calculate throughput
                if ( secs>=timeslots){

                    for (int j=1;j<=timeslots;j++){
                       sum=sum+packpersec[secs-timeslots+j];
                    }
                //      System.out.println(sum);
                    throughput[sessioncount]= sum*q.getLength()*8/timeslots;
                 //   System.out.println(throughput[sessioncount]);
                    trp.println(throughput[sessioncount]);
```

```
                    sessioncount+=1;
                    sum=0;
                }

             //    System.out.println(q.getLength());
                  secs+=1;
                  numofpack=0;
              }


            //  Thread.sleep(500);
                  }
                  catch( Exception e ){
                          System.out.println("Failed to fetch data package after 4sec");
          }
              send.send(f);
                          i=System.currentTimeMillis();
                  }

                  output.close();

          }
          catch (Exception x) {
                          System.out.println(x);}

  }
  //--------------------------------------------------------------------------
  public void image(int camera) {
          PrintStream outa = null;
          String packetInfoimage1;
          //int camera=1;

          packetInfoimage1 =packetInfoimage+"CAM="+camera+"FLOW=ON";
          txbuffer = packetInfoimage.getBytes();
          img = new DatagramPacket(txbuffer,txbuffer.length, hostAddress,serverPort);

          try {
                  OutputStream image = new FileOutputStream("image1.jpg");


           send.send(img);
           img = new DatagramPacket("NEXT".getBytes(),"NEXT".length(), hostAddress,serverPort);

           for (;;){
                   try{
           receive.receive(q);
          //String message = new String(rxbuffer,0,q.getLength());

           byte[] imageArray = Arrays.copyOfRange(q.getData(),0,q.getLength());
           image.write(imageArray);
           System.out.println(imageArray);

           Thread.sleep(400);
                   }
                   catch (Exception e) {
                                   System.out.println(e);
                                   break;
                                   }
           send.send(img);

              }
          }
           catch (Exception x) {
                  System.out.println(x);
                  }

  }
  //--------------------------------------------------------------------------
  public void getTemp() {
          String packetInfo1;

  try{

          FileWriter temperature = new FileWriter("temperature.txt");
```

```
    BufferedWriter output = new BufferedWriter(temperature);

        for (int i=1;i<8;++i){

                packetInfo1=packetInfoecho+"T0"+i;
                txbuffer = packetInfo1.getBytes();
                f = new DatagramPacket(txbuffer,txbuffer.length, hostAddress,serverPort);
                send.send(f);

                receive.receive(q);
                String message = new String(rxbuffer,0,q.getLength());
                System.out.println(message);
                if (message.length()>32){
                output.write(message,27,19);
                output.newLine();
                }
                Thread.sleep(500);
         }
        output.close();

    }
catch (Exception x) {
        System.out.println(x);
        }

}
//---------------------------------------------------------------------------------------
public void Audio( int packages, String modulation) {

        int Q=16;
        try{
                AudioFormat linearPCM = new AudioFormat(8000,Q,1,true,false);
                //linearPCM einai antikeimeno morfopoihshs hxou

                SourceDataLine lineOut = AudioSystem.getSourceDataLine(linearPCM);
                //lineout einai i eksodos hxou

                //energopoihsh eksodou
                //audiobuffer size = 32000
                lineOut.open(linearPCM,32000);


                lineOut.start();
                if (modulation.contains("AQ")){
                   lineOut.write(this.getaqdpcm(packages),0,this.getaqdpcm(packages).length);
                }
                else{

                   lineOut.write(this.getdpcm(packages),0,this.getdpcm(packages).length);
                }

                lineOut.stop();
                lineOut.close();
        }

        catch (Exception x) {
                System.out.println(x);
                }


}


//---------------------------------------------------------------------------
public byte[] getdpcm(int packages){
        String packetInfo1;

        int SP=packages;
    byte[] buff = new byte[ 128 * 2 ];
    byte[] audio = new byte[ SP * 2 * 128 ];
    int counter = 0;
    int nibble = 0;
    int pack=0;
    int song=1;
```

```
    packetInfo1 = packetInfoaudio+"F"+SP;
        txbuffer = packetInfo1.getBytes();
        f = new DatagramPacket(txbuffer,txbuffer.length, hostAddress,serverPort);

    try{
        FileWriter fw = new FileWriter( "DPCMT" + ".txt" );
        BufferedWriter bw = new BufferedWriter( fw );
        send.send( f );

        for( int j = 0; j < SP; ++j ){
            try{
                receive.receive( q );
               pack++;
                buff = q.getData();
                System.out.println(buff);
                for( int i = 0; i < 128; ++i ){
                    int X1 = ( buff[ i ] >> 4 ) & 0x0f;
                    int X2 = buff[ i ] & 0x0f;

                    X1 = X1 - 8;
                    X2 = X2 - 8;

                    X1 += nibble;
                    if( X1 > 127 ){
                        X1 = 127;
                    }
                    if( X1 < -128 ){
                        X1 = -128;
                    }
                    X2 += X1;
                    if( X2 > 127 ){
                        X2 = 127;
                    }
                    if( X2 < -128 ){
                        X2 = -128;
                    }
                    nibble = X2;

                    byte x1 = ( byte ) X1;
                    byte x2 = ( byte ) X2;

                    audio[ counter++ ] = x1;
                    audio[ counter++ ] = x2;

                    bw.write( X1 + " " );
                    bw.write( X2 + " " );
                }
            }
            catch( Exception x ){
                break;
            }
        }
        bw.close();
   //     System.out.println(pack);
    }
    catch( Exception x ){
        System.out.println(x );
    }
    return audio;
}
public byte[] getaqdpcm(int packages){

        String packetInfo1;
        int SP=packages;
    byte[] buff = new byte[ 132 * 2 ];
    byte[] audio = new byte[ SP * 4 * 128 ];
    int counter = 0;
    int nibble = 0;
    int song=1;

    packetInfo1 = packetInfoaudio+"F"+SP;
        txbuffer = packetInfo1.getBytes();
        f = new DatagramPacket(txbuffer,txbuffer.length, hostAddress,serverPort);
```

```
    try{
        FileWriter fw1 = new FileWriter( "AQDPCM28" + ".txt" );
        BufferedWriter log = new BufferedWriter( fw1 );
        FileWriter fw = new FileWriter( "AQDPCMmusic28" + ".txt" );
        BufferedWriter bw = new BufferedWriter( fw );
        send.send( f );

        for( int j = 0; j < SP; ++j ){
            try{
                receive.receive( q );
                buff = q.getData();
                System.out.println(buff);
                byte[] bb = new byte[ 4 ];
                byte sign = (byte)( ( buff[ 1 ] & 0x80 ) != 0 ? 0xff : 0x00 );
                bb[ 3 ] = sign;
                bb[ 2 ] = sign;
                bb[ 1 ] = buff[ 1 ];
                bb[ 0 ] = buff[ 0 ];

                int m = ByteBuffer.wrap( bb ).order( ByteOrder.LITTLE_ENDIAN ).getInt();


                sign = (byte)( ( buff[ 3 ] & 0x80 ) != 0 ? 0xff : 0x00 );
                bb[ 3 ] = sign;
                bb[ 2 ] = sign;
                bb[ 1 ] = buff[ 3 ];
                bb[ 0 ] = buff[ 2 ];

                int b = ByteBuffer.wrap( bb ).order( ByteOrder.LITTLE_ENDIAN ).getInt();
                log.write( m + " " + b + "\n" );
                for( int i = 4; i < 132; ++i ){
                    int D1 = ( buff[ i ] >>> 4 ) & 0x0f;
                    int D2 = buff[ i ] & 0x0f;

                    int d1 = D1 - 8;
                    int d2 = D2 - 8;

                    int delta1 = d1 * b;
                    int delta2 = d2 * b;

                    int X1 = delta1 + nibble;
                    int X2 = delta2 + delta1;
                    nibble = delta2;

                    int x1 = X1 + m;
                    int x2 = X2 + m;

                    audio[ counter++ ] = ( byte ) ( x1 );
                    audio[ counter++ ] = ( byte ) ( x1 / 256 > 127 ? 127 : x1 / 256 < -128 ? -
128 : x1 / 256 );
                    audio[ counter++ ] = ( byte ) ( x2 );
                    audio[ counter++ ] = ( byte ) ( x2 / 256 > 127 ? 127 : x2 / 256 < -128 ? -
128 : x2 / 256 );

                    bw.write( x1 + " " );
                    bw.write( x2 + " " );
                }

            }
            catch( Exception x ){
                break;
            }
        }
        bw.close();
        log.close();
    }
    catch( Exception x ){
        System.out.println( x );
    }
    return audio;
}
//--------------------------------------------------------------------------
}
```