Δίκτυα Υπολογιστών 2
Σωκράτης Κοσέογλου 8837 sokrkose@ece.auth.gr

```java
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.Scanner;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.SourceDataLine;
import java.awt.Desktop;
import java.io.*;

public class UserApplication {

  int clientPort = 48009;
  int serverPort = 38009;

  String echoPayload = "E1136";
  String echoPayloadE0000 = "E0000";
  String imagePayload = "M0907";
  String audioPayload = "A9519";
  String ithakiCopterPayload = "Q9986";
  String vehiclePayload = "V8190"; //Need to be changed inside the method too

  byte[] hostIP = { (byte) 155, (byte) 207, (byte) 18, (byte) 208 };

  public static void main(String[] args) throws IOException {

    System.out.println("Do you want to run Standalone Packets or Session ? [1, 2]
");
    System.out.println("1. Standalone Packets");
    System.out.println("2. Session");
    Scanner input1 = new Scanner(System.in);
    int option = input1.nextInt();

    while (option != 1 && option != 2) {
      System.out.println("Wrong Number...Press again!");
      Scanner input2 = new Scanner(System.in);
      option = input2.nextInt();
    }

    if (option == 1){
      System.out.println("Enter mode of operation: [1, 2, 3, 4, 5]");
      System.out.println("1. Echo Packet");
      System.out.println("2. Image Packet");
      System.out.println("3. Audio Packet");
      System.out.println("4. IthakiCopter Packet");
      System.out.println("5. Vehicle Packet");
      Scanner input3 = new Scanner(System.in);
      int mode = input3.nextInt();

      while (mode < 1 || mode > 5) {
        System.out.println("Wrong Number...Press again!");
        Scanner input4 = new Scanner(System.in);
        mode = input4.nextInt();
      }

      if (mode == 1) {
        (new UserApplication()).echo(0, 1, 1);        //echo(long echoPacketsSeco
nds, int packet, int temp)
      } else if (mode == 2) {
        (new UserApplication()).image(0, "PTZ");      //image(int imagePacketLeng
th, String camera)
      } else if (mode == 3) {
        (new UserApplication()).audio(null, 0, 0, 8); //audio(String encodingMode
, int soundMode, int numberOfPackets, int song)
      } else if (mode == 4) {
        (new UserApplication()).ithakiCopter(0);      //ithakiCopter(long ithakiC
opterSeconds)
      } else {
        (new UserApplication()).vehicle(0);           //vehicle(long vehicleSecon
ds)
      }

    }else{


      // SESSION MODE !!!

      (new UserApplication()).echo(250, 1, 0);
      (new UserApplication()).throughput(8, "../../../../Desktop/Δικτυα 2/Εργασια
/SESSION2/responsetimes250.csv", "../../../../Desktop/Δικτυα 2/Εργασια/SESSION2/t
hroughputs250.csv");
      (new UserApplication()).echo(250, 0, 0);
      (new UserApplication()).throughput(8, "../../../../Desktop/Δικτυα 2/Εργασια
/SESSION2/responsetimesE0000250.csv", "../../../../Desktop/Δικτυα 2/Εργασια/SESSI
ON2/throughputsE0000250.csv");
```

```
      //CHANGE ECHO OUTPUT FILE !!!
      (new UserApplication()).echo(250, 1, 1);

      (new UserApplication()).echo(5, 1, 0);
      (new UserApplication()).image(1024, "FIX");
      (new UserApplication()).echo(5, 1, 0);
      (new UserApplication()).image(1024, "PTZ");

      (new UserApplication()).echo(5, 1, 0);
      (new UserApplication()).audio("DPCM", 1, 999, 8);
      (new UserApplication()).echo(5, 1, 0);
      (new UserApplication()).audio("DPCM", 2, 999, 8);
      (new UserApplication()).echo(5, 1, 0);
      (new UserApplication()).audio("AQ-DPCM", 2, 999, 8);
      // CHANGE FILES !!!
      (new UserApplication()).echo(5, 1, 0);
      (new UserApplication()).audio("AQ-DPCM", 2, 999, 7);

      (new UserApplication()).echo(5, 1, 0);
      (new UserApplication()).ithakiCopter(250);
      // CHANGE FILES !!!
      (new UserApplication()).echo(5, 1, 0);
      (new UserApplication()).ithakiCopter(250);

      (new UserApplication()).echo(5, 1, 0);
      (new UserApplication()).vehicle(250);

    }

  }

  public void echo(long echoPacketsSeconds, int packet, int temp) throws IOExcept
ion {

    if (packet == 0){
      echoPayload = echoPayloadE0000;
    }

    if (temp == 1){
      echoPayload = echoPayload + "T00";
    }

    byte[] echoBuffer = echoPayload.getBytes();

    if (echoPacketsSeconds == 0){
      System.out.println("For how many seconds do you want to send packets?");
      Scanner input1 = new Scanner(System.in);
      echoPacketsSeconds = input1.nextInt();

      while (echoPacketsSeconds <= 0) {
        System.out.println("Wrong Number...Press again!");
        Scanner input2 = new Scanner(System.in);
        echoPacketsSeconds = input2.nextInt();
      }
    }

    long startTime = 0, endTime = 0, elapsedTime = 0, sendTime = 0, receiveTime =
 0, delta = 0;

    PrintWriter responseTimesLog = null, temperatures = null;

    if (packet == 0){
      responseTimesLog = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SE
SSION2/responsetimesE0000250.csv");
    }else if (packet == 1){
      responseTimesLog = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SE
SSION2/responsetimes250.csv");
    }

    if (temp == 1){
      temperatures = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSIO
N2/Temperatures.csv");
    }

    DatagramSocket mySendSocket = new DatagramSocket();
    InetAddress hostAddress = InetAddress.getByAddress(hostIP);
    DatagramPacket sendPacket = new DatagramPacket(echoBuffer, echoBuffer.length,
 hostAddress, serverPort);

    DatagramSocket myRecieveSocket = new DatagramSocket(clientPort);
    myRecieveSocket.setSoTimeout(4000);
    byte[] rxbuffer = new byte[50];
    DatagramPacket recievePacket = new DatagramPacket(rxbuffer, rxbuffer.length);

    mySendSocket.send(sendPacket);
    startTime = System.currentTimeMillis();
    sendTime = System.currentTimeMillis();
    int T;

    while ((elapsedTime / 1000) < echoPacketsSeconds) {
      try {
        myRecieveSocket.receive(recievePacket);

        receiveTime = System.currentTimeMillis();
        delta = receiveTime - sendTime;
        responseTimesLog.println(delta);

        String message = new String(rxbuffer, 0, recievePacket.getLength());
        System.out.println(message);

        if (temp == 1){
          T = Integer.parseInt(message.substring(44, 46), 10);
          temperatures.println(T);
        }

      } catch (Exception x) {
        System.out.println(x);
        responseTimesLog.println((long ) 0);
      }

      endTime = System.currentTimeMillis();
      elapsedTime = endTime - startTime;
      mySendSocket.send(sendPacket);
      sendTime = System.currentTimeMillis();
    }

    System.out.println("");
    System.out.println("We recieved echo pachets for " + (double) elapsedTime / 1
000 + " seconds");

    if (temp == 1){
      temperatures.close();
    }
    responseTimesLog.close();
    mySendSocket.close();
    myRecieveSocket.close();
  }

  public void throughput(int interval, String inputFile, String outputFile) throw
s IOException {

    ArrayList<String> responseTimesString = new ArrayList<String>(Files.readAllLi
nes(Paths.get(inputFile)));
    ArrayList<Long> responseTimesLong = new ArrayList<Long>();
    for (String k : responseTimesString){
      responseTimesLong.add(Long.parseLong(k, 10));
      System.out.println(Long.parseLong(k, 10));
    }

    ArrayList<Float> throughputs = new ArrayList<Float>();
    ArrayList<Long> summary = new ArrayList<Long>();

    long sum = 0;
    for (int i = 0; i < responseTimesLong.size(); i++) {
      for (int j = 0; j <= i; j++) {
        sum += responseTimesLong.get(j);
      }
      summary.add(sum);
      sum = 0;
    }

    float packetspersecond = 0;
    long lower = 0;
    long upper = 1000 * interval;

    while (upper < summary.get(summary.size() - 1) + 1000) {
      for (int j = 0; j < summary.size(); j++) {
        if ((summary.get(j) > lower) && (summary.get(j) < upper)) {
          packetspersecond++;
        }
      }
      lower += 1000;
      upper += 1000;
      throughputs.add(packetspersecond / interval);
      packetspersecond = 0;
    }

    PrintWriter log = new PrintWriter(outputFile, "UTF-8");

    for (float var : throughputs) {
      log.println(var);
      System.out.println(var);
    }

    log.close();
  }

  public void image(int imagePacketLength, String camera) throws IOException {
    if (imagePacketLength == 0){
      System.out.println("Choose one of the following image packet lenght: [128, 
256, 512, 1024]");
      System.out.println("1. 128 bytes");
      System.out.println("2. 256 bytes");
      System.out.println("3. 512 bytes");
      System.out.println("4. 1024 bytes");
      Scanner input1 = new Scanner(System.in);
      imagePacketLength = input1.nextInt();

      while (imagePacketLength != 128 && imagePacketLength != 256 && imagePacketL
ength != 512
          && imagePacketLength != 1024) {
        System.out.println("Wrong Number...Press again!");
        Scanner input2 = new Scanner(System.in);
        imagePacketLength = input2.nextInt();
      }
    }

    if(camera == "FIX"){
      imagePayload = imagePayload + "CAM=" + camera;
    }else if(camera == "PTZ"){
      imagePayload = imagePayload + "CAM=" + camera;
    }

    imagePayload = imagePayload + "UDP=" + imagePacketLength;
    byte[] imageBuffer = imagePayload.getBytes();

    DatagramSocket mySendSocket = new DatagramSocket();
    InetAddress hostAddress = InetAddress.getByAddress(hostIP);
    DatagramPacket sendPacket = new DatagramPacket(imageBuffer, imageBuffer.lengt
h, hostAddress, serverPort);

    DatagramSocket myRecieveSocket = new DatagramSocket(clientPort);
    myRecieveSocket.setSoTimeout(2000);
    byte[] rxbuffer = new byte[1024];
    DatagramPacket recievePacket = new DatagramPacket(rxbuffer, rxbuffer.length);

    mySendSocket.send(sendPacket);


    File file = null;
    FileOutputStream imageFile = null;
    if (camera == "FIX"){
      file = new File("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2/imageFIX.png
");
      imageFile = new FileOutputStream("../../../../Desktop/Δικτυα 2/Εργασια/SESS
ION2/imageFIX.png");
    }else if (camera == "PTZ"){
      file = new File("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2/imagePTZ.png
");
      imageFile = new FileOutputStream("../../../../Desktop/Δικτυα 2/Εργασια/SESS
ION2/imagePTZ.png");
    }

    int payload = imagePacketLength;

    while (payload == imagePacketLength) {

      try {
        myRecieveSocket.receive(recievePacket);
        payload = recievePacket.getLength();
        for (int i = 0; i < imagePacketLength; i++) {
          imageFile.write(rxbuffer[i]);
        }
      } catch (Exception x) {
        System.out.println(x);
      }

    }

    Desktop desktop = Desktop.getDesktop();
    desktop.open(file);

    imageFile.close();
    mySendSocket.close();
    myRecieveSocket.close();

  }

  public void audio(String encodingMode, int soundMode, int numberOfPackets, int
song) throws IOException {

    if (encodingMode == null) {
      System.out.println("Choose one of the following audio modulations: [1,2]");
      System.out.println("1. DPCM");
      System.out.println("2. AQ-DPCM");
      Scanner input1 = new Scanner(System.in);
      int in1 = input1.nextInt();
      if(in1 == 1){
        encodingMode = "DPCM";
      }else if(in1 == 2){
        encodingMode = "AQ-DPCM";
      }

      while ((encodingMode != "DPCM") && (encodingMode != "AQ-DPCM")) {
        System.out.println("Wrong Number...Press again!");
        Scanner input2 = new Scanner(System.in);
        int in2 = input2.nextInt();
        if(in2 == 1){
          encodingMode = "DPCM";
        }else if(in2 == 2){
          encodingMode = "AQ-DPCM";
        }
      }
    }

    if (soundMode == 0) {
      System.out.println("Choose one of the following audio packet mode: [1,2]");
      System.out.println("1. Sine wave");
      System.out.println("2. Audio Clip");
      Scanner input3 = new Scanner(System.in);
      soundMode = input3.nextInt();

      while (soundMode != 1 && soundMode != 2) {
        System.out.println("Wrong Number...Press again!");
        Scanner input4 = new Scanner(System.in);
        soundMode = input4.nextInt();
      }
    }

    if (numberOfPackets == 0) {
      System.out.println("How many audio packets do you want? [000-999]");
      Scanner input5 = new Scanner(System.in);
      numberOfPackets = input5.nextInt();

      while (numberOfPackets < 0 || numberOfPackets > 999) {
        System.out.println("Wrong Number...Press again!");
        Scanner input6 = new Scanner(System.in);
        numberOfPackets = input6.nextInt();
      }
    }
    if (encodingMode == "DPCM"){
        if (soundMode == 1){
          audioPayload = audioPayload + "T" + numberOfPackets;
        }else{
          if (song > 0){
            audioPayload = audioPayload + "L0" + String.valueOf(song) + "F" + num
berOfPackets;
          }else{
            audioPayload = audioPayload + "F" + numberOfPackets;
          }
        }
    }else{
      audioPayload = audioPayload + "AQ";
      if (soundMode == 1){
        audioPayload = audioPayload + "T" + numberOfPackets;
      }else{
        if (song > 0){
          audioPayload = audioPayload + "L0" + String.valueOf(song) + "F" + numbe
rOfPackets;
        }else{
          audioPayload = audioPayload + "F" + numberOfPackets;
        }
      }
    }


    int overhead = 0, Q = 8, buffersize = 1, b = 1, mean = 0;

    byte[] audioBuffer = audioPayload.getBytes();
    DatagramSocket mySendSocket = new DatagramSocket();
    InetAddress hostAddress = InetAddress.getByAddress(hostIP);
    DatagramPacket sendPacket = new DatagramPacket(audioBuffer, audioBuffer.lengt
h, hostAddress, serverPort);

    DatagramSocket myRecieveSocket = new DatagramSocket(clientPort);
    myRecieveSocket.setSoTimeout(3000);
    byte[] rxbuffer = new byte[128 + overhead];
    DatagramPacket recievePacket = new DatagramPacket(rxbuffer, rxbuffer.length);

    if(encodingMode == "AQ-DPCM"){
      overhead = 4;
      Q = 16;
      buffersize = 2;
    }
    int bLSB, bMSB, bAQ, mLSB, mMSB, mAQ;

    int[] nibblesamples = new int[256];
    int nibble1, nibble2, diff1, diff2, sample0 = 0, sample1 = 0, sample2 = 0;

    byte[] audioBufferOut = new byte[buffersize * numberOfPackets * 256];
    int[] demux = new int[256];

    AudioFormat linearPCM = new AudioFormat(8000, Q, 1, true, false);
    SourceDataLine lineOut = null;

    try {
      lineOut = AudioSystem.getSourceDataLine(linearPCM);
      lineOut.open(linearPCM, 256*numberOfPackets);
      lineOut.start();
    } catch (LineUnavailableException x) {
      System.out.println(x);
    }

    PrintWriter meanfile = null, stepfile = null, samples = null, dpcmfreq = null
, diffsDPCM = null, diffsAQ = null;

    try {
      if (encodingMode == "DPCM") {
        if (soundMode == 1) {
          diffsDPCM = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSI
ON2/DPCMdiffs_SIN.csv", "UTF-8");
          dpcmfreq = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSIO
N2/DPCMfreq_SIN.csv", "UTF-8");
        } else {
          diffsDPCM = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSI
ON2/DPCMdiffs_Clip.csv", "UTF-8");
          dpcmfreq = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSIO
N2/DPCMfreq_Clip.csv", "UTF-8");
        }
      }else{
        meanfile = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2
/AQDPCMmeanfile2.csv", "UTF-8");
        diffsAQ = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2/
AQDPCMdiffs2.csv", "UTF-8");
        stepfile = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2
/AQDPCMstepfile2.csv", "UTF-8");
        samples = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2/
AQDPCMsamples2.csv", "UTF-8");
      }
    } catch (Exception x) {
      System.out.println(x);
    }

    mySendSocket.send(sendPacket);

    for(int j=0; j<numberOfPackets; j++){

      try{

        myRecieveSocket.receive(recievePacket);

        bLSB = (int ) (rxbuffer[2] & 0xFF);
        bMSB = (int ) (rxbuffer[3] & 0xFF);
        bAQ = bMSB * 256 + bLSB;

        mLSB = (int) (rxbuffer[0] & 0xFF);
        mMSB = (int) (rxbuffer[1]);
        mAQ = mMSB * 256 + mLSB;

        if (encodingMode == "AQ-DPCM") {
          mean = mAQ;
          b = bAQ;
        }

        for(int i=0 + overhead; i < rxbuffer.length; i++){

          nibble1 = (byte ) ((rxbuffer[i] & 240) >> 4);
          nibble2 = (byte) (rxbuffer[i] & 15);

          nibblesamples[i - overhead] = (int) nibble1;
          nibblesamples[i + 1 - overhead] = (int) nibble2;

          diff1 = (nibblesamples[i - overhead] - 8) * b;
          diff2 = (nibblesamples[i + 1 - overhead] - 8) * b;

          if (encodingMode == "DPCM") {
            sample1 = diff1 + sample0;
            sample2 = diff2 + sample1;
            sample0 = sample2;

            dpcmfreq.println(sample1);
            dpcmfreq.println(sample2);
            diffsDPCM.println((int) nibble1 - 8);
            diffsDPCM.println((int) nibble2 - 8);
          }else{
            sample1 = sample0 + diff1 + mean;
            sample2 = diff1 + diff2 + mean;
            sample0 = diff2;

            samples.println(sample1);
            samples.println(sample2);
            diffsAQ.println((int) nibble1 - 8);
            diffsAQ.println((int) nibble2 - 8);
          }

          demux[(i - overhead) * 2] = sample1;
          demux[(i - overhead) * 2 + 1] = sample2;

        }

        if (encodingMode == "DPCM") {
          for (int i = 0; i < rxbuffer.length; i++) {
            audioBufferOut[256 * j + i * 2] = (byte) demux[i * 2];
            audioBufferOut[256 * j + i * 2 + 1] = (byte) demux[i * 2 + 1];
          }
        }else{
          for (int i = 0; i < rxbuffer.length - 4; i++) {
            audioBufferOut[512 * j + i * 4] = (byte) (demux[i * 2] & 0xFF);
            audioBufferOut[512 * j + i * 4 + 1] = (byte) ((demux[i * 2] >> 8) & 0
xFF);
            audioBufferOut[512 * j + i * 4 + 2] = (byte) (demux[i * 2 + 1] & 0xFF
);
            audioBufferOut[512 * j + i * 4 + 3] = (byte) ((demux[i * 2 + 1] >> 8)
 & 0xFF);
          }

          meanfile.println(mean);
          stepfile.println(b);

        }

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

    if (encodingMode == "AQ-DPCM") {
      meanfile.close();
      stepfile.close();
      samples.close();
      diffsAQ.close();
    }else{
      dpcmfreq.close();
      diffsDPCM.close();
    }

    lineOut.write(audioBufferOut, 0, buffersize * 256 * numberOfPackets);
    lineOut.stop();
    lineOut.close();
    mySendSocket.close();
    myRecieveSocket.close();
  }

  public void ithakiCopter(long ithakiCopterSeconds) throws IOException {

    int clientIthakiCopterPort = 48078;
    int serverIthakiCopterPort = 38078;

    if (ithakiCopterSeconds == 0){
      System.out.println("For how many seconds do you want to operate the ithakiC
opter ?");
      Scanner input1 = new Scanner(System.in);
      ithakiCopterSeconds = input1.nextInt();

      while (ithakiCopterSeconds <= 0) {
        System.out.println("Wrong Number...Press again!");
        Scanner input2 = new Scanner(System.in);
        ithakiCopterSeconds = input2.nextInt();
      }
    }

    PrintWriter leftMotor = null, rightMotor = null, altitude = null, temperature
 = null, pressure = null;
    String lMotor, rMotor, alt, temp, press;

    try{
      leftMotor = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2/
leftMotor1.csv");
      rightMotor = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2
/rightMotor1.csv");
      altitude = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2/a
ltitude1.csv");
```

```
      temperature = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2/temperature1.csv");
      pressure = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2/pressure1.csv");
    } catch (Exception x){
      System.out.println(x);
    }

    byte[] ithakiCopterBuffer = ithakiCopterPayload.getBytes();

    DatagramSocket mySendSocket = new DatagramSocket();
    InetAddress hostAddress = InetAddress.getByAddress(hostIP);
    DatagramPacket sendPacket = new DatagramPacket(ithakiCopterBuffer, ithakiCopterBuffer.length, hostAddress, serverIthakiCopterPort);

    DatagramSocket myRecieveSocket = new DatagramSocket(clientIthakiCopterPort);
    myRecieveSocket.setSoTimeout(4000);
    byte[] rxbuffer = new byte[5000];
    DatagramPacket recievePacket = new DatagramPacket(rxbuffer, rxbuffer.length);

    long startTime = System.currentTimeMillis();
    long elapsedTime = 0, endTime;

    while (elapsedTime/1000 < ithakiCopterSeconds){
      try{
        mySendSocket.send(sendPacket);
        myRecieveSocket.receive(recievePacket);
        String message = new String(rxbuffer, 0, recievePacket.getLength());
        System.out.println(message);

        lMotor = message.substring(40, 43);
        rMotor = message.substring(51, 54);
        alt = message.substring(64, 67);
        temp = message.substring(80, 86);
        press = message.substring(96, 103);

        leftMotor.println(lMotor);
        rightMotor.println(rMotor);
        altitude.println(alt);
        temperature.println(temp);
        pressure.println(press);

      } catch (IOException x){
        System.out.println(x);
      }
      endTime = System.currentTimeMillis();
      elapsedTime = endTime - startTime;
    }

    leftMotor.close();
    rightMotor.close();
    altitude.close();
    temperature.close();
    pressure.close();
    mySendSocket.close();
    myRecieveSocket.close();
  }

  public void vehicle(long vehicleSeconds) throws IOException {

    if (vehicleSeconds== 0){
      System.out.println("For how many seconds do you want to take data from the
vehicle ?");
      Scanner input1 = new Scanner(System.in);
      vehicleSeconds = input1.nextInt();

      while (vehicleSeconds <= 0) {
        System.out.println("Wrong Number of Seconds...Press again!");
        Scanner input2 = new Scanner(System.in);
        vehicleSeconds = input2.nextInt();
      }
    }

    PrintWriter logFile_1F = null;
    PrintWriter logFile_0F = null;
    PrintWriter logFile_11 = null;
    PrintWriter logFile_0C = null;
    PrintWriter logFile_0D = null;
    PrintWriter logFile_05 = null;

    try {
      logFile_1F = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2
/vehicleLog_1F.csv", "UTF-8");
      logFile_0F = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2
/vehicleLog_0F.csv", "UTF-8");
      logFile_11 = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2
/vehicleLog_11.csv", "UTF-8");
      logFile_0C = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2
/vehicleLog_0C.csv", "UTF-8");
```

```
      logFile_0D = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2
/vehicleLog_0D.csv", "UTF-8");
      logFile_05 = new PrintWriter("../../../../Desktop/Δικτυα 2/Εργασια/SESSION2
/vehicleLog_05.csv", "UTF-8");

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

    for(int pid = 0; pid < 6; pid++){

      String vehiclePayload = "V8190";
      String[] pidCodes = { "1F", "0F", "11", "0C", "0D", "05" };

      DatagramSocket mySendSocket = null;
      DatagramSocket myRecieveSocket = null;

      vehiclePayload = vehiclePayload + "OBD=01 " + pidCodes[pid];
      byte[] vehicleBuffer = vehiclePayload.getBytes();

      mySendSocket = new DatagramSocket();
      InetAddress hostAddress = InetAddress.getByAddress(hostIP);
      DatagramPacket sendPacket = new DatagramPacket(vehicleBuffer, vehicleBuffer
.length, hostAddress, serverPort);

      myRecieveSocket = new DatagramSocket(clientPort);
      myRecieveSocket.setSoTimeout(4000);
      byte[] rxbuffer = new byte[1024];
      DatagramPacket recievePacket = new DatagramPacket(rxbuffer, rxbuffer.length
);

      long startTime = System.currentTimeMillis();
      long elapsedTime = 0, endTime;
      int xx, yy, equation = 0;

      while (elapsedTime/1000 < vehicleSeconds){
        try{
          mySendSocket.send(sendPacket);
          myRecieveSocket.receive(recievePacket);
          String message = new String(rxbuffer, 0, recievePacket.getLength());
          System.out.println(message);

          xx = Integer.parseInt(message.substring(6, 8), 16);

          if (pid == 0){
```

```
          yy = Integer.parseInt(message.substring(9, 11), 16);
          equation = ((256 * xx) + yy);
          logFile_1F.println(equation);
        }else if (pid == 1){
          equation = (xx - 40);
          logFile_0F.println(equation);
        }else if (pid == 2){
          equation = ((xx * 100) / 255);
          logFile_11.println(equation);
        }else if (pid == 3){
          yy = Integer.parseInt(message.substring(9, 11), 16);
          equation = (((xx * 256) + yy) / 4);
          logFile_0C.println(equation);
        }else if (pid == 4){
          equation = xx;
          logFile_0D.println(equation);
        }else if (pid == 5){
          equation = (xx - 40);
          logFile_05.println(equation);
        }

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

      endTime = System.currentTimeMillis();
      elapsedTime = endTime - startTime;
    }

    if (pid == 0){
      logFile_1F.close();
    }else if (pid == 1){
      logFile_0F.close();
    }else if (pid == 2){
      logFile_11.close();
    }else if (pid == 3){
      logFile_0C.close();
    }else if (pid == 4){
      logFile_0D.close();
    }else if (pid == 5){
      logFile_05.close();
    }

    mySendSocket.close();
    myRecieveSocket.close();

    }

  }

}
```