```
//Diktya II     Aristoteleio Panepisthmio Thesssalonikhs
//Mastoras Rafail Evagelos 7918  6 Apriliou 2017

import java.net.*;
import java.io.*;
import java.util.Scanner;
import java.util.ArrayList;
import javax.sound.sampled.*;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.text.SimpleDateFormat;
import java.util.Calendar;



public class userApplication {
 public static void main(String[] param) {
          (new userApplication()).menu();
 }


 public void menu(){
   Scanner input = new Scanner(System.in);
   int mode=0;
   int serverPort=0 ;
   System.out.print("Enter the server listening port for this session: ");
   serverPort = Integer.parseInt(input.nextLine());

   int clientPort=0 ;
   System.out.print("Enter the client listening port for this session: ");
   clientPort = Integer.parseInt(input.nextLine());
   int echoCode=0,imageCode=0,audioCode=0,copterCode=0,vehicleCode=0;

   for(;;){
     System.out.print("\n1. Echo Request");
     System.out.print("\n2. Image Request\n3. Sound Request (DPCM)\n4. Sound Request
(AQDPCM)\n5. Ithakicopter\n6. Vehicle ODB-II\n7. Exit\n");
     try{
       int select=Integer.parseInt(input.nextLine());

       if (select==1){

         if(echoCode==0){
           System.out.print("Enter the echo request code EXXXX for this session: ");
           echoCode = Integer.parseInt(input.nextLine());
         }

         System.out.print("\nChoose mode: \n");
         System.out.print("1.Echo with delay\n");
         System.out.print("2.Echo without delay\n");
         mode = Integer.parseInt(input.nextLine());

         echo(echoCode,mode,serverPort,clientPort);
```

```
}else if(select==2){
 if(imageCode==0){
   System.out.print("Enter the image request code MXXXX for this session: ");
   imageCode = Integer.parseInt(input.nextLine());
 }

 System.out.print("\nChoose mode: \n");
 System.out.print("1.Cam 1\n");
 System.out.print("2.Cam 2\n");
 mode = Integer.parseInt(input.nextLine());

 image(imageCode,mode,serverPort,clientPort);
}else if(select==3){
   if(audioCode==0){
     System.out.print("Enter the audio request code AXXXX for this session: ");
     audioCode = Integer.parseInt(input.nextLine());
   }

                    System.out.print("\nChoose mode:\n");
                    System.out.print("1.Song DPCM\n");
                    System.out.print("2.Frequency DPCM\n");
                    mode = Integer.parseInt(input.nextLine());
  soundDPCM(audioCode,mode,serverPort,clientPort);
}else if(select==4){
   if(audioCode==0){
     System.out.print("Enter the audio request code AXXXX for this session: ");
     audioCode = Integer.parseInt(input.nextLine());
   }

   System.out.print("\nChoose mode:\n");
   System.out.print("1.Song DPCM\n");
   System.out.print("2.Frequency DPCM\n");
   mode = Integer.parseInt(input.nextLine());
   soundAQDPCM(audioCode,mode,serverPort,clientPort);
}else if(select==5){
 if(copterCode==0){
   System.out.print("Enter the Ithakicopter request code QXXXX for this session: ");
   copterCode = Integer.parseInt(input.nextLine());
 }
  Ithakicopter(copterCode,38038,48038);
}else if(select==6){
 if(vehicleCode==0){
   System.out.print("Enter the Vehicle ODB-II request code VXXXX for this session: ");
   vehicleCode = Integer.parseInt(input.nextLine());
 }
 String[] pid= new String[6];
 mode=0;
 pid[0]="1F";
 pid[1]="0F";
 pid[2]="11";
 pid[3]="0C";
 pid[4]="0D";
 pid[5]="05";
```

```
        System.out.print("\nChoose pid:\n");
        System.out.print("0.Engine Run Time\n");
        System.out.print("1.Intake air temperature\n");
        System.out.print("2.Throttle possition\n");
        System.out.print("3.Engine RPM\n");
        System.out.print("4.Vehicle Speed\n");
        System.out.print("5.Coolant temperature\n");

        mode = Integer.parseInt(input.nextLine());
        if(mode>5){
         System.out.print("\nError :You choose a non valid option:\n");
         return;
        }
        System.out.println("vehicle code= "+vehicleCode);
        vehicle(vehicleCode,serverPort,clientPort,pid[mode]);


      }else if(select==7){
        System.out.print("Programm exit! \n");
        return;

      }else{
        System.out.print("You have selected none of the listed options, try again \n\n");
      }
    }catch(Exception x){
         System.out.print("You have selected none of the listed options, try again \n\n");
    }
  }
 }

 public void echo(int echoCode,int mode,int serverPort,int clientPort) throws
SocketException,IOException,UnknownHostException{
   String message="",modeinfo="";
   String packetInfo ="";
   if(mode==1){
     modeinfo="withDelay";
     packetInfo="E" + Integer.toString(echoCode) +"\r";
   }else{
     modeinfo="withoutDelay";
     packetInfo="E0000\r";
   }


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

   byte[] txbuffer = packetInfo.getBytes();

   DatagramSocket sendSocket = new DatagramSocket();
   DatagramPacket sendPacket = new DatagramPacket(txbuffer,txbuffer.length,
hostAddress,serverPort);

   DatagramSocket recieveSocket = new DatagramSocket(clientPort);

   recieveSocket.setSoTimeout(3600);
   byte[] rxbuffer = new byte[2048];
   DatagramPacket recievePacket = new DatagramPacket(rxbuffer,rxbuffer.length);

   ArrayList<String> filewrite = new ArrayList<String>();
   ArrayList<Double> packetTime = new ArrayList<Double>();
   ArrayList<Float> counters = new ArrayList<Float>();
   int numberOfPackets=0;
   //data input from server
   double startTime=0,endTime=0,totalTime=0,avg=0;
   double startloop=0,endloop=0;
   startloop = System.nanoTime();

   while(endloop<(5*60*1000)){
   sendSocket.send(sendPacket);
   numberOfPackets++;
   startTime = System.nanoTime();
   for (;;) {
     try {
        recieveSocket.receive(recievePacket);
        endTime=(System.nanoTime()- startTime)/1000000;
        message = new String(rxbuffer,0,recievePacket.getLength());
        System.out.println(message);
        System.out.print("Time to recieve this packet was "+endTime+"\n");
        break;
     } catch (Exception x) {
      System.out.println(x);
      break;
     }
   }


   filewrite.add("Time to recieve this packet was "+endTime+"\n");
   packetTime.add(endTime);

   totalTime+=endTime;
   endloop=(System.nanoTime()-startloop)/1000000;

 }


 avg=totalTime/numberOfPackets;
 filewrite.add("Number of packets : "+String.valueOf((double)numberOfPackets));
 filewrite.add("Average time : "+String.valueOf(avg));
 filewrite.add("Total time of communication : "+(totalTime/60)/1000+" minutes\n");
 filewrite.add("Total time of test : "+(endloop/60)/1000+" minutes\n");
double sumInterval=0;
float counterInterval=0;

 for(int i = 0; i < packetTime.size();i++){
   int j = i;
   while((sumInterval < 8*1000)&&(j < packetTime.size())){
     sumInterval += packetTime.get(j);
     counterInterval++;
     j++;
   }
   counterInterval = counterInterval/8;
   counters.add(counterInterval);
   counterInterval = 0;
   sumInterval = 0;
 }

 BufferedWriter bw = null;
 try{
   File file =new File(("Echo"+echoCode+modeinfo+".txt"));
   bw = new BufferedWriter(new FileWriter(("Echo"+echoCode+modeinfo+".txt"), false));
   if(!file.exists()){
     file.createNewFile();
   }
   for (int i=0; i <filewrite.size(); i++){

     bw.write(String.valueOf(filewrite.get(i)));
     bw.newLine();
   }
   bw.newLine();
 }catch(IOException ioe){
   ioe.printStackTrace();
 }
 finally{
   try{
     if(bw != null) bw.close();
   }catch(Exception ex){
     System.out.println("Error in closing the BufferedWriter" + ex);
   }
 }

 bw = null;
 try{
   File file =new File(("PacketsPer8Seconds"+echoCode+modeinfo+".txt"));
   bw = new BufferedWriter(new FileWriter(("PacketsPer8Seconds"+echoCode+modeinfo+".txt"),
false));

   if(!file.exists()){
     file.createNewFile();
   }
   for (int i=0; i <counters.size(); i++){

     bw.write(String.valueOf(counters.get(i)));
     bw.newLine();
   }
   bw.newLine();
 }catch(IOException ioe){
   ioe.printStackTrace();
 }
 finally{
   try{
     if(bw != null) bw.close();
   }catch(Exception ex){
     System.out.println("Error in closing the BufferedWriter" + ex);
   }
```

```
}


System.out.println("Now sending for temperatures \n");
startloop = System.nanoTime();
for(int i=0;i<=9;i++){
packetInfo="E"+Integer.toString(echoCode)+"T0"+i+"\r";
txbuffer = packetInfo.getBytes();
sendPacket = new DatagramPacket(txbuffer,txbuffer.length, hostAddress,serverPort);
sendSocket.send(sendPacket);
numberOfPackets++;
startTime = System.nanoTime();
for (;;) {
  try {
      recieveSocket.receive(recievePacket);
      endTime=(System.nanoTime()- startTime)/1000000;
      message = new String(rxbuffer,0,recievePacket.getLength());
      System.out.println(message);
      System.out.print("Time to recieve this packet was "+endTime+"\n");
      break;
  } catch (Exception x) {
    System.out.println(x);
    break;
  }
}
totalTime+=endTime;
endloop=(System.nanoTime()-startloop)/1000000;

}






recieveSocket.close();
sendSocket.close();

System.out.print("Average packet recieve time:"+avg+" milliseconds for "+numberOfPackets+ "
packets\n");

}

public void image(int imageCode,int mode,int serverPort,int clientPort) throws
SocketException,IOException,UnknownHostException{
     String packetInfo="",modeinfo="";
     if(mode == 1){
        modeinfo="CAM1";
        packetInfo = "M" + Integer.toString(imageCode) +"\r";
     }else if(mode == 2){
        modeinfo="CAM2";
        packetInfo = "M" + Integer.toString(imageCode) + " " + "CAM=PTZ" + "\r";
     }

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

     byte[] txbuffer = packetInfo.getBytes();

     DatagramSocket sendSocket = new DatagramSocket();
     DatagramPacket sendPacket = new DatagramPacket(txbuffer,txbuffer.length,
hostAddress,serverPort);

     DatagramSocket recieveSocket = new DatagramSocket(clientPort);

     recieveSocket.setSoTimeout(3600);

     byte[] rxbuffer = new byte[2048];
     DatagramPacket recievePacket = new DatagramPacket(rxbuffer,rxbuffer.length);


     sendSocket.send(sendPacket);
                 recieveSocket.setSoTimeout(3200);
     String fileName = ("image"+imageCode+modeinfo+".jpeg");
                 FileOutputStream outputStream = new FileOutputStream(fileName);
                 for(;;){
                         try{
                                 recieveSocket.receive(recievePacket);
                                 if (rxbuffer == null) break;
                                 for(int i = 0 ; i <= 127 ; i++){
                                 outputStream.write(rxbuffer[i]);
                                 }
                         }catch (IOException ex) {
                                 System.out.println(ex);
                                 break;
                         }
                 }
                 outputStream.close();

     System.out.print("Image saving ended\n");
     recieveSocket.close();
     sendSocket.close();

}

public void soundDPCM(int audioCode,int mode,int serverPort,int clientPort) throws
SocketException,IOException,UnknownHostException,LineUnavailableException{
  int numPackets = 999,mask1 = 15,mask2 = 240,b = 5,rx;
  int soundSample1 = 0,soundSample2 = 0;
  int nibble1,nibble2,sub1,sub2,x1 = 0,x2 = 0,counter = 0;
  String message = "",packetInfo = "",modeinfo="";
  ArrayList<Integer> subs = new ArrayList<Integer>();
  ArrayList<Integer> samples = new ArrayList<Integer>();




  if(mode == 1){
    modeinfo="song";
    packetInfo = "A" + Integer.toString(audioCode) + "F999";
```

```
  }else if(mode == 2){
    modeinfo="freq";
    packetInfo = "A" + Integer.toString(audioCode) + "T999";
  }

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

  byte[] txbuffer = packetInfo.getBytes();

  DatagramSocket sendSocket = new DatagramSocket();
  DatagramPacket sendPacket = new DatagramPacket(txbuffer,txbuffer.length,
hostAddress,serverPort);

  DatagramSocket recieveSocket = new DatagramSocket(clientPort);

  recieveSocket.setSoTimeout(3600);

  byte[] rxbuffer = new byte[128];
  DatagramPacket recievePacket = new DatagramPacket(rxbuffer,rxbuffer.length);

  sendSocket.send(sendPacket);

    byte[] song = new byte[256*numPackets];
    for(int i = 1;i < numPackets;i++){
      try{
        recieveSocket.receive(recievePacket);
        for (int j = 0;j <= 127;j++){
          rx = rxbuffer[j];
          nibble1 = rx & mask1; //mask1 00001111
          nibble2 = (rx & mask2)>>4; //mask2 11110000
          sub1 = nibble1-8;
          subs.add(sub1);
          sub1 = sub1*b;
          sub2 = nibble2-8;
          subs.add(sub2);
          sub2 = sub2*b;
          x1 = x2 + sub1;
          samples.add(x1);
          x2 = x1 + sub2;
          samples.add(x2);
          song[counter] = (byte)x1;
          song[counter + 1] = (byte)x2;
          counter += 2;
        }
    }catch (Exception ex){
      System.out.println(ex);
    }
    if((i%100)==0){
      System.out.println((1000-i)+" samples left");
    }

  }

  if(mode==1){
```

```
  System.out.println("Playing the song");

  AudioFormat pcm = new AudioFormat(8000,8,1,true,false);
  SourceDataLine playsong = AudioSystem.getSourceDataLine(pcm);
  playsong.open(pcm,32000);
  playsong.start();
  playsong.write(song,0,256*numPackets);
  playsong.stop();
  playsong.close();
}


BufferedWriter bw = null;
try{
  File file = new File("DPCMsubF"+audioCode+modeinfo+".txt");

  if(!file.exists()){
    file.createNewFile();
  }
  FileWriter fw = new FileWriter(file,false);
  bw = new BufferedWriter(fw);
  for(int i = 0 ; i < subs.size() ; i += 2){
    bw.write("" + subs.get(i) + " " + subs.get(i+1));
    bw.newLine();
  }

}catch(IOException ioe){
  ioe.printStackTrace();
}finally{
  try{
    if(bw != null) bw.close();
  }catch(Exception ex){
    System.out.println("Error in closing the BufferedWriter" + ex);
  }
}

BufferedWriter mw = null;
try{
  File file = new File("DPCMsamplesF"+audioCode+modeinfo+".txt");
  if(!file.exists()){
    file.createNewFile();
  }
  FileWriter fw = new FileWriter(file,false);
  mw = new BufferedWriter(fw);
  for(int i = 0 ; i < samples.size() ; i += 2){
    mw.write("" + samples.get(i) + " " + samples.get(i+1));
    mw.newLine();
  }

}catch(IOException ioe){
  ioe.printStackTrace();
}finally{
  try{
    if(mw != null) mw.close();
  }catch(Exception ex){
      System.out.println("Error in closing the BufferedWriter" + ex);
    }
  }

  recieveSocket.close();
  sendSocket.close();


 }

 public void soundAQDPCM(int audioCode,int mode,int serverPort,int clientPort) throws
SocketException,IOException,UnknownHostException,LineUnavailableException{
  int numPackets = 999,mask1 = 15,mask2 = 240,rx;
  int soundSample1 = 0,soundSample2 = 0;
  int nibble1,nibble2,sub1,sub2,x1 = 0,x2 = 0,counter = 4,mean,b,temp = 0;
  String message = "",packetInfo = "",modeinfo="";
  ArrayList<Integer> means = new ArrayList<Integer>();
  ArrayList<Integer> bs = new ArrayList<Integer>();
  ArrayList<Integer> subs = new ArrayList<Integer>();
  ArrayList<Integer> samples = new ArrayList<Integer>();




  if(mode == 1){
    modeinfo="song";
    packetInfo = "A" + Integer.toString(audioCode) + "AQF999";
  }else if(mode == 2){
    modeinfo="freq";
    packetInfo = "A" + Integer.toString(audioCode) + "AQT999";
  }

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

  byte[] txbuffer = packetInfo.getBytes();

  DatagramSocket sendSocket = new DatagramSocket();
  DatagramPacket sendPacket = new DatagramPacket(txbuffer,txbuffer.length,
hostAddress,serverPort);

  DatagramSocket recieveSocket = new DatagramSocket(clientPort);

  byte[] rxbuffer = new byte[132];
  DatagramPacket recievePacket = new DatagramPacket(rxbuffer,rxbuffer.length);
  recieveSocket.setSoTimeout(5000);
  sendSocket.send(sendPacket);
    byte[] meanByte = new byte[4];
    byte[] bByte = new byte[4];
    byte sign;
    byte[] song = new byte[256*2*numPackets];
    for(int i = 1;i < numPackets;i++){
      if((i%100)==0){
        System.out.println((1000-i)+" samples left");
      }
```

```
     try{
       recieveSocket.receive(recievePacket);
       sign = (byte)( ( rxbuffer[1] & 0x80) !=0 ? 0xff : 0x00);//if rxbuffer[1]&10000000=0 then sign =0
else =01111111 , we take the compliment of this number
       meanByte[3] = sign;
       meanByte[2] = sign;
       meanByte[1] = rxbuffer[1];
       meanByte[0] = rxbuffer[0];
       mean = ByteBuffer.wrap(meanByte).order(ByteOrder.LITTLE_ENDIAN).getInt(); //convert the
array into integer number using LITTLE_ENDIAN format
       means.add(mean);
       sign = (byte)( ( rxbuffer[3] & 0x80) !=0 ? 0xff : 0x00);
       bByte[3] = sign;
       bByte[2] = sign;
       bByte[1] = rxbuffer[3];
       bByte[0] = rxbuffer[2];
       b = ByteBuffer.wrap(bByte).order(ByteOrder.LITTLE_ENDIAN).getInt();
       bs.add(b);

       for (int j = 4;j <= 131;j++){ //the remaining bytes are the samples
         rx = rxbuffer[j];
         nibble1 = (int)(rx & 0x0000000F);
         nibble2 = (int)((rxbuffer[j] & 0x000000F0)>>4);
         sub1 = (nibble2-8);
         subs.add(sub1);
         sub2 = (nibble1-8);
         subs.add(sub2);
         sub1 = sub1*b;
         sub2 = sub2*b;
         x1 = temp + sub1 + mean;
         samples.add(x1);
         x2 = sub1 + sub2 + mean;
         temp = sub2;
         samples.add(x2);
         counter += 4;
         song[counter] = (byte)(x1 & 0x000000FF);
         song[counter + 1] = (byte)((x1 & 0x0000FF00)>>8);
         song[counter + 2] = (byte)(x2 & 0x000000FF);
         song[counter + 3] = (byte)((x2 & 0x0000FF00)>>8);


       }
    }catch (Exception ex){
      System.out.println(ex);
    }
  }
  if(mode==1){
    System.out.println("Playing the song");

    AudioFormat aqpcm = new AudioFormat(8000,16,1,true,false);
    SourceDataLine playsong = AudioSystem.getSourceDataLine(aqpcm);
    playsong.open(aqpcm,32000);
    playsong.start();
    playsong.write(song,0,256*2*numPackets);
    playsong.stop();
    playsong.close();
  }


  BufferedWriter bw = null;
  try{
    File file = new File("AQDPCMsubsF"+audioCode+modeinfo+".txt");
    if(!file.exists()){
      file.createNewFile();
    }
    FileWriter fw = new FileWriter(file,false);
    bw = new BufferedWriter(fw);
    for(int i = 0 ; i < subs.size() ; i += 2){
      bw.write("" + subs.get(i) + " " + subs.get(i+1));
      bw.newLine();
    }

  }catch(IOException ioe){
    ioe.printStackTrace();
  }finally{
    try{
      if(bw != null) bw.close();
    }catch(Exception ex){
      System.out.println("Error in closing the BufferedWriter" + ex);
    }
  }

  BufferedWriter mw = null;
  try{
    File file = new File("AQDPCMsamplesF"+audioCode+modeinfo+".txt");
    if(!file.exists()){
      file.createNewFile();
    }
    FileWriter fw = new FileWriter(file,false);
    mw = new BufferedWriter(fw);
    for(int i = 0 ; i < samples.size() ; i += 2){
      mw.write("" + samples.get(i) + " " + samples.get(i+1));
      mw.newLine();
    }

  }catch(IOException ioe){
    ioe.printStackTrace();
  }finally{
    try{
      if(mw != null) mw.close();
    }catch(Exception ex){
      System.out.println("Error in closing the BufferedWriter" + ex);
    }
  }

  BufferedWriter pw = null;
  try{
    File file = new File("AQDPCMmeanF"+audioCode+modeinfo+".txt");
    if(!file.exists()){
      file.createNewFile();
    }
    FileWriter fw = new FileWriter(file,false);
    pw = new BufferedWriter(fw);
    for(int i = 0 ; i < means.size() ; i += 2){
      pw.write("" + means.get(i));
      pw.newLine();
    }

  }catch(IOException ioe){
    ioe.printStackTrace();
  }finally{
    try{
      if(pw != null) pw.close();
    }catch(Exception ex){
      System.out.println("Error in closing the BufferedWriter" + ex);
    }
  }

  BufferedWriter kw = null;
  try{
    File file = new File("AQDPCMbetasF"+audioCode+modeinfo+".txt");
    if(!file.exists()){
      file.createNewFile();
    }
    FileWriter fw = new FileWriter(file,false);
    kw = new BufferedWriter(fw);
    for(int i = 0 ; i < bs.size() ; i ++){
      kw.write("" + bs.get(i));
      kw.newLine();
    }

  }catch(IOException ioe){
    ioe.printStackTrace();
  }finally{
    try{
      if(kw != null) kw.close();
    }catch(Exception ex){
      System.out.println("Error in closing the BufferedWriter" + ex);
    }
  }

  recieveSocket.close();
  sendSocket.close();

 }

 public void Ithakicopter(int copterCode,int serverPort,int clientPort) throws
SocketException,IOException,UnknownHostException,LineUnavailableException,ClassNotFoundExce
ption{
  String packetInfo="",message="";
  ArrayList<String> messages = new ArrayList<String>();

  packetInfo = "Q" + Integer.toString(copterCode)+"\r";
  byte[] hostIP = { (byte)155,(byte)207,18,(byte)208 };
  InetAddress hostAddress = InetAddress.getByAddress(hostIP);
  byte[] txbuffer = packetInfo.getBytes();
  DatagramSocket sendSocket = new DatagramSocket();
  DatagramPacket sendPacket = new DatagramPacket(txbuffer,txbuffer.length,
hostAddress,serverPort);
  DatagramSocket recieveSocket = new DatagramSocket(clientPort);
  byte[] rxbuffer = new byte[5000];
  DatagramPacket recievePacket = new DatagramPacket(rxbuffer,rxbuffer.length);
  Calendar cal = Calendar.getInstance();
  SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
  messages.add("Current Session "+copterCode+" Current Time "+sdf.format(cal.getTime())+"\n");
  recieveSocket.setSoTimeout(5000);
  for (int i = 1;i <= 60 ; i++){
   try{
     sendSocket.send(sendPacket);
                recieveSocket.receive(recievePacket);
                message = new String(rxbuffer,0,recievePacket.getLength());
                messages.add(message);
                System.out.println(message);
   }catch(Exception ex){
    System.out.println(ex);
   }

        }

        BufferedWriter bw = null;
                try{
                        File file = new File("Ithakicopter"+copterCode+".txt");
                        if(!file.exists()){
                                file.createNewFile();
                        }
                        FileWriter fw = new FileWriter(file,true);
                        bw = new BufferedWriter(fw);
                        for(int i = 0 ; i < messages.size(); i++){
                                bw.write("" + messages.get(i));
                                bw.newLine();
                        }

                }catch(IOException ioe){
                        ioe.printStackTrace();
                }finally{
                        try{
                                if(bw != null) bw.close();
                        }catch(Exception ex){
                                System.out.println("Error in closing the BufferedWriter" + ex);
                        }
                }

   recieveSocket.close();
   sendSocket.close();
 }

 public void vehicle(int vehicleCode,int serverPort,int clientPort,String pid) throws
SocketException,IOException,UnknownHostException,LineUnavailableException,ClassNotFoundExce
ption{
  String packetInfo="",message="";
  ArrayList<String> messages = new ArrayList<String>();
  double startloop=0,endloop=0;


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

  DatagramSocket sendSocket = new DatagramSocket();
  DatagramSocket recieveSocket = new DatagramSocket(clientPort);
  byte[] rxbuffer = new byte[5000];
  DatagramPacket recievePacket = new DatagramPacket(rxbuffer,rxbuffer.length);

   recieveSocket.setSoTimeout(5000);
   startloop = System.nanoTime();
   while(endloop<4*60*1000){

      packetInfo = "V"+Integer.toString(vehicleCode)+"OBD=01 "+pid+"\r";
      byte[] txbuffer = packetInfo.getBytes();
      DatagramPacket sendPacket = new DatagramPacket(txbuffer,txbuffer.length,
hostAddress,serverPort);
      try{
        sendSocket.send(sendPacket);
                recieveSocket.receive(recievePacket);
                message = new String(rxbuffer,0,recievePacket.getLength());
                messages.add(message);
                System.out.println(message);
      }catch(Exception ex){
        System.out.println(ex);
      }


    endloop=(System.nanoTime()-startloop)/1000000;
   }

        BufferedWriter bw = null;
                try{
                        File file = new File("OBDVehicle"+vehicleCode+"PID"+pid+".txt");
                        if(!file.exists()){
                                file.createNewFile();
                        }
                        FileWriter fw = new FileWriter(file,true);
                        bw = new BufferedWriter(fw);
                        for(int i = 0 ; i < messages.size(); i++){
                                bw.write("" + messages.get(i));
                                bw.newLine();
                        }

                }catch(IOException ioe){
                        ioe.printStackTrace();
                }finally{
                        try{
                                if(bw != null) bw.close();
                        }catch(Exception ex){
                                System.out.println("Error in closing the BufferedWriter" + ex);
```

```
                              }
                      }

  recieveSocket.close();
  sendSocket.close();
 }


}//userApplication class
```