```java
package com.antonis;

import ithakimodem.Modem;

import java.io.*;
import java.util.ArrayList;

/*
 *
 * Δίκτυα Υπολογιστών I
 *
 * Experimental Virtual Lab
 *
 * Java virtual modem communications seed code
 *
 */

public class application {
    public static void main(String[] param) throws FileNotFoundException {
        //Initialize modem object
        Modem modem;
        modem = new Modem(12000);
        modem.setTimeout(2000);

        //Run desired functions of application, passing the modem object along with the code as args.
        (new application()).demo(modem, "ATD2310ITHAKI\r");
        (new application()).arq(modem, "Q9790", "R4647"); //Pass without the \r, the function does it
        (new application()).echo(modem, "E7720");
        (new application()).image(modem, "M5110", "goodImage.jpg");
        (new application()).image(modem, "G3012", "badImage.jpg");
        (new application()).gps(modem, "P8332");

        //Gracefully close the connection
        modem.close();
    }

    public void demo(Modem modem, String address){
        int k;
        modem.write(address.getBytes());
        for (; ; ) {
            try {
                k = modem.read();
                if (k == -1) break;
                System.out.print((char) k);
            } catch (Exception x) {
               break;
            }
        }
    }
```

```
    public void echo(Modem modem, String echo_code) {
        echo_code += appendCarriageReturn();
        int k;//Temp variable
        ArrayList<Integer> responseTimes = new ArrayList<>(); //ArrayList containing response time valu
es to store in csv
        String message = "";

        //Variables to keep track of time
        double tick;
        double tock = 0;
        double startTime = System.currentTimeMillis(); //Start of communication

        //Loop that runs for 240.000 ms or 4 minutes
        while (tock - startTime < 241000) {
            modem.write(echo_code.getBytes());
            tick = System.currentTimeMillis(); //Time before reading a new line
            for (;;) {
                try {
                    k = modem.read();
                    if (k == -1) {
                        break;
                    }
                    System.out.print((char) k);
                    message += (char) k;
                    if(message.contains("PSTART ")){
                        message = "";
                        tick = System.currentTimeMillis();
                    }
                    if(message.contains(" PSTOP")){
                        tock = System.currentTimeMillis();
                        message = "";
                        break;
                    }
                } catch (Exception e) {
                    break;
                }
            }
            int delay = (int) (tock - tick); //Calculate response time in milliseconds

            responseTimes.add(delay); //Add to ArrayList
            System.out.println("\n\tResponse Time: " + (delay) + " ms"); //Print some stats
            System.out.println("\tTime to end: " + (240 - (tock - startTime) / 1000) + " seconds. \n");
        }
        //File Write try catch block
        try {
            FileWriter fw = new FileWriter("ping.csv");
            BufferedWriter bw = new BufferedWriter(fw);
            for (Integer responseTime : responseTimes) {
                bw.write(responseTime.toString() + ','); //Write every item of responseTimes arraylist
to a buffer, separated by commas
            }
```

```
            bw.close();
            fw.close();
            System.out.println("\nWrote csv table. \n");
        } catch (IOException e) {
            System.out.println("Something went wrong");
        }
    }

    //255, 216 means start of image
    //255, 217 means end of image
    public void image(Modem modem, String imageCode, String imageName) throws FileNotFoundException {
        modem.setSpeed(80000);
        imageCode += appendCarriageReturn(); //Codes passed do not have "\r", so I'm adding it here
        File image = new File(imageName);
        FileOutputStream imageStream = new FileOutputStream(image); //Create a new file to which we can
 stream the bytes we get sent
        int currentValue, lastValue; //I'm reading two bytes at a time, since the important byte patter
s for an image are two bytes long
        boolean writingImage = false; //Shows if I'm in the middle of an image write
        System.out.print("\nRequesting " + imageName);
        try {
            modem.write(imageCode.getBytes()); //Request the appropriate image by its code
            for(;;) {
                lastValue = modem.read();
                currentValue = modem.read(); //Read two bytes at a time
                if(lastValue == 255 && currentValue == 216 || writingImage) { //If the two bytes match
the start sequence OR writing has already started, write to file and set status
                    imageStream.write(lastValue);
                    imageStream.write(currentValue);
                    writingImage = true;
                }
                if (lastValue==-1) { //If I should have exited 1 byte ago, exit without writing
                    break;
                }
                if (currentValue==-1){ //If I should exit now, write the last byte and exit
                    imageStream.write(lastValue);
                    break;
                }
                if(lastValue==255 && currentValue==217) { //If both bytes match the ending sequence, wr
ite them to file and exit
                    imageStream.write(lastValue);
                    imageStream.write(currentValue);
                    break;
                }
            }
            imageStream.flush();
            imageStream.close();
            System.out.println("\nWrote " + imageName + "\n");
        }
        catch (Exception e) {
            System.out.println("Something went wrong");
```

```
    }
}

public void gps(Modem modem, String gpsCode) throws FileNotFoundException {
    int k;
    String initialRequestCode = gpsCode + "R=1011150" + appendCarriageReturn(); //Modify the access
 code to give initial packets
    StringBuilder newGpsCode = new StringBuilder(gpsCode); //Access code to give gps image
    modem.write(initialRequestCode.getBytes()); //Request initial packets
    StringBuilder message = new StringBuilder(); //Write all incoming characters to a huge string
    for (; ;) {
        try {
            k = modem.read();
            if (k == -1) break;
            System.out.print((char) k);
            message.append((char) k);
            if(message.toString().contains("STOP ITHAKI GPS TRACKING")){
                break;
            }
        } catch (Exception x) {
            System.out.println("Something went wrong");
        }
    }
    String[] packets = message.toString().split("\r\n"); //Split by new line, making every item of
the array a separate packet
    String[] cleanPackets = new String[packets.length - 2];
    //Clean packets removes START ITHAKI GPS TRACKING and STOP ITHAKI GPS TRACKING items
    System.arraycopy(packets, 1, cleanPackets, 0, cleanPackets.length); //an item of cleanPackets w
ill look like this
    //$GPGGA,103750.000,4037.7705,N,02257.5465,E,1,08,1.1,48.6,M,36.1,M,,0000*6F

    String latDeg; //if my latitude is e.g. 4037.7705 i need to separate 7705 and convert
it to seconds
    String latSec; //im taking the 7705 part, multiplying by 0.006 to return the seconds and append
ing it to my gps image request code
    String longDeg; //same for longitude
    String longSec;
    int latSecConvert; //e.g. 7705 * 0.006 will be this
    int longSecConvert; //same for longitude

    for(String s : cleanPackets){
        latDeg = (s.split(",")[2]).split("\\.")[0]; //first split keeps this 4037.7705 and second s
plit keeps 4037
        latSec = (s.split(",")[2]).split("\\.")[1]; //first split keeps this 4037.7705 and second s
plit keeps 7705
        longDeg = (s.split(",")[4]).split("\\.")[0].substring(1); //first split keeps 02257.5465, s
econd split keeps 02257 and substring keeps 2257
        longSec = (s.split(",")[4]).split("\\.")[1]; //first split keeps 02257.5465, second split k
eeps 5465
        latSecConvert = (int) (Integer.parseInt(latSec) * 0.006); //convert what should be seconds
to seconds
```

```
            longSecConvert = (int) (Integer.parseInt(longSec) * 0.006); //same for longitude
            newGpsCode.append("T=").append(longDeg).append(longSecConvert).append(latDeg).append(latSec
Convert); //create new gps code by adding all the coordinates
        }
        image(modem, newGpsCode.toString(), "gps.jpg"); //request an image with the correct code

    }
    public void arq(Modem modem, String ack_code, String nack_code) {
        int k; //Temp variable
        ack_code += appendCarriageReturn(); //Add "\r" to codes
        nack_code += appendCarriageReturn();
        String message; //A message is a string that contains the whole packet
        e.g. PSTART 28-03-2021 11:56:36 23 <aYIlQqjnOQopRhXg> 061 PSTOP
        String payload; //Payload keeps only e.g. aYIlQqjnOQopRhXg
        int checksum; //Is the checksum I should be getting from ITHAKI e.g. 61
        boolean lastCorrect = true; //Shows if my last receiving packet was correct
        int acks = 0; //Total ACKs
        int nacks = 0; //Total NACKs
        int nack_retries = 0; //Attempts required to get the correct transmission for a single packet
        ArrayList<Integer> responseTimes = new ArrayList<>(); //Response times until a correct packet
        ArrayList<Integer> retryList = new ArrayList<>(); //Attempts to get a correct packet

        //Variables to keep track of time
        double tick = 0;
        double tock = 0;
        double startTime = System.currentTimeMillis(); //Start of communication

        //Loop that runs for 240.000 ms or 4 minutes
        while (tock - startTime < 241000) {
            message = "";
            int packetsum = 0; //Is the result of the XOR operation between all the characters of paylo
ad
            if(lastCorrect){
                modem.write(ack_code.getBytes()); //If my last packet was received correctly, request a
 new one
            }
            else{
                nack_retries++; //If my last packet wasn't received correctly, add 1 to attempts counte
r
                modem.write(nack_code.getBytes()); //Request the same packet again
            }
            for (;;) {
                try {
                    k = modem.read();
                    if (k == -1) {
                        break;
                    }
                    System.out.print((char) k);
                    message += ((char) k); //Read the packet and add it to a String
                    if(message.contains("PSTART")){
                        message = "";
```

```
                    tick = System.currentTimeMillis(); //Time before reading a new line
                }
                if(message.contains("PSTOP")){
                    break;
                }
            } catch (Exception e) {
                System.out.println("Something went wrong");
            }
        }
        message = message.split(" PSTOP")[0]; //Split at " PSTOP" and keep the left part e.g. PSTAR
T 28-03-2021 11:56:36 23 <aYIlQqjnOQopRhXg> 061
        payload = message.split("<")[1]; //Split at "<" and keep the right part e.g. aYIlQqjnOQopRh
Xg> 061
        payload = payload.split("> ")[0]; //Split at "> " and keep the left part e.g. aYIlQqjnOQopR
hXg
        checksum = Integer.parseInt(message.split("> ")[1]); //Split at "> " and keep the right par
t e.g. 061, then parse the integer giving you 61 as an int
        for (char c : payload.toCharArray()){
            packetsum = packetsum^(int) c; //For every character in payload, do the XOR operation s
equentially with the result of every previous operation
        }
        if(packetsum == checksum){ //If the checksums match
            tock = System.currentTimeMillis(); //Time after reading a new line
            lastCorrect = true; //Set the correct flag to true
            retryList.add(nack_retries); //Write the number of attempts to get a correct package to
 the ArrayList
            nack_retries = 0; //Set the new number of attempts to 0
            int delay = (int) (tock - tick); //Calculate response time in milliseconds
            System.out.println("\n\tResponse Time: " + (delay) + " ms");
            responseTimes.add(delay); //Add response time to ArrayList
            System.out.println("\tTime to end: " + (240 - (tock - startTime) / 1000) + " seconds. \
n");
            acks++; //Add 1 to the total ACKs counter
        }
        else{
            lastCorrect = false; //Set the correct flag to false
            nacks++; //Add 1 to the total NACKs counter
        }
        System.out.println("\nACKS: " + acks); //Print some stats
        System.out.println("NACKS: " + nacks + "\n\n");
    }
    try { //Write to files, one with the responseTimes and one with RetryList as .csv
        FileWriter fwPing = new FileWriter("arq_ping.csv");
        BufferedWriter bwPing = new BufferedWriter(fwPing);
        for (Integer responseTime : responseTimes) {
            bwPing.write(responseTime.toString() + ','); //Write every item of responseTimes arrayl
ist to a buffer, separated by commas
        }
        bwPing.close();
        fwPing.close();
        System.out.println("\nWrote ARQ ping csv table. \n");
```

```
            FileWriter fwRetry = new FileWriter("arq_retries.csv");
            BufferedWriter bwRetry = new BufferedWriter(fwRetry);
            for (Integer attempt : retryList) {
                bwRetry.write(attempt.toString() + ','); //Write every item of retryList arraylist to a
buffer, separated by commas
            }
            bwRetry.close();
            fwRetry.close();
            System.out.println("\nWrote ARQ retry csv table. \n");
        } catch (IOException e) {
            System.out.println("Something went wrong");
        }
    }

    public String appendCarriageReturn(){   return "\r";    }
    //Helper function to add "\r" to modem codes, making them appropriate to send to Ithaki
}
```