Δίκτυα Υπολογιστών 1

~ Source ~

Παππάς Δημήτριος

8391

Ανάλυση Κώδικα:

Αρχικά, θα γίνει ανάλυση και επεξήγηση μεμονομένων τμημάτων κώδικα. Ενώ στη συνέχεια, θα γίνει παράθεση του πηγαίου κώδικα.

Αρχικοποίηση τιμών:

- Δίνεται τιμή στα request codes
- Ορίζουμε στα 80Kbps την ταχύτητα επικοινωνίας
- Ορίζεται η τιμή του timeout στα 2 δευτερόλεπτα

```
12
13      Modem modem  = new Modem();
14      modem.open("ithaki");
15      modem.setSpeed(80000);                              // 80 Kbps
16      modem.setTimeout(2000);
17
18      String temp = new String("");
19      String buffer = new String("");
20      int k;
21
22      String echo = new String("E8185\r");
23      String image_error_free = new String("M2680\r");
24      String image_with_error = new String("G7471\r");
25      String gps = new String("P9202R=1000090\r");        // R=XPPPPLL  -  X = presaved route (1)  -  PPPP = starting position (0000
26      String gps_image_code = new String("P9202");        // follows code T=AABBCCDDEEZZ\r
27      String ack = new String("Q7721\r");
28      String nack = new String("R3491\r");
29
```

Σύνδεση στο server:

- Το virtual modem της Ithaki στέλνει μήνυμα κατά τη σύνδεση του χρήστη
- Εκτυπώνουμε το μήνυμα (προαιρετικά)

```
32      // Log In
33      System.out.println("Hello ithaki!\n");
34      for (;;){
35          try {
36              k = modem.read();
37              if (k==-1){
38                  break;
39              }if(k!=-1){
40                  // do nothing
41              }
42              // System.out.print((char)k);
43          }catch (Exception x){
44              break;
45          }
46      }
47
```

Echo Πακέτα:

- Στέλνουμε συνεχόμενα request για αποστολή echo πακέτων από την Ithaki
- Η λήψη ενός πακέτου σταματάει μόλις λάβουμε την έκφραση “PSTOP”
- Υπολογίζουμε το χρόνο αποστολής κάθε πακέτου
- Η διαδικασία τερματίζει μόλις περάσουν 5 λεπτά
- Αποθηκεύουμε σε file τους χρόνους απόκρισης κάθε πακέτου

```
52        // ECHO PACKAGE REQUEST   -   REMAKE
53        System.out.println("Request echo package");
54        int counter = 1;                                          // number of packages
55        long start = System.currentTimeMillis();                  // keep the current time in msec when requesting a package
56        long start_of_5_minites = start;                          // keep the current time in msec when writing to modem for the first time
57        long end;                                                 // keep the current time in msec when package arrived
58        long passed;                                              // time passed until package received
59        long threshold;                                           // time passed since first request
60        ArrayList<String> pack = new ArrayList<String>();         // store packages
61        ArrayList<Long> time = new ArrayList<Long>();             // store response time
62
63        // Get echo packages
64        for (;;) {
65            try{
66                buffer = "";
67                modem.write(echo.getBytes());                     // request packages from server
68                start = System.currentTimeMillis();
69                for (;;) {
70                    k = modem.read();                             // PSTART DD-MM-YYYY HH-MM-SS PC PSTOP
71                    buffer = buffer + (char)(k);
72                    if (buffer.contains("PSTOP")) {               // package received
73                        end = System.currentTimeMillis();
74                        passed = end - start;
75                        pack.add(buffer);
76                        time.add(passed);
77                        // System.out.println(buffer);
78                        break;
79                    }
80                }
81                threshold = end - start_of_5_minites;
82                counter++;
83                if (threshold>1000*60*5){                             // 5 minutes in msec for echo packages request duration
84                    System.out.println("5 minutes for echo packages request duration have passed");
85                    break;
86                }
87                continue;
88            }catch (Exception x) {
89                break;
90            }
91        }
92
93        System.out.println("number of echo packages = " + counter);
94
94
95         try {
96             FileWriter writer = new FileWriter("Response_Times.txt", true); // file writer
97             for (int i=0 ; i<time.size() ; i++) {
98                 writer.write(Long.toString(time.get(i)));           // write response time in a file
99                 temp = "\n";                                        // press enter
100                writer.write(temp);
101            }
102            writer.close();
103         }catch (IOException e) {
104            System.out.println("An error occurred");
105            e.printStackTrace();
106        }
107        System.out.println("Echo package finished");
108
```

Εικόνα χωρίς σφάλμα:

- Γίνεται request εικόνας χωρίς σφάλμα
- Διατηρούμε τις default ρυθμίσεις για την λήψη της εικόνας
- Ελέγχουμε τα δύο πρώτα και δύο τελευταία bytes της εικόνας, ώστε να καταλάβουμε αν έχει σταλθεί σωστά
- Αποθηκεύουμε την εικόνα ως binary αρχείο jpg

```
113         // IMAGE ERROR FREE REQUEST
114         System.out.println("Request image error free package");
115         modem.write(image_error_free.getBytes());                   // request image
116         ArrayList<Integer> image = new ArrayList<Integer>();        // Store image
117
118         // Get image
119         for (;;) {
120             try {
121                 k = modem.read();
122                 if (k==-1) {
123                     // finished reading
124                     break;
125                 }
126                 image.add(k);                                       // store package in image list
127             }catch (Exception x) {
128                 break;
129             }
130         }
131
132         // Image received is stored in array
133         int[] frame = new int[image.size()];                        // store image in array
134         for (int i=0 ; i<image.size() ; i++) {
135             frame[i] = image.get(i);
136         }
137
138         // check if image is correct
139         if (frame[0]==0xFF && frame[1]==0xD8 && frame[image.size()-2]==0xFF && frame[image.size()-1]==0xD9) {
140             System.out.println("Image is correct");
141         }else {
142             System.out.println("Image is NOT correct  -  Abort mission");
143         }
144
145         // Write image in file
146         FileOutputStream output = null;                             // FileOutputStream is meant for writing streams of raw bytes such as
147         try {
148             output = new FileOutputStream("image_error_free.jpg");
149             for (int i=0 ; i<image.size() ; i++) {
150                 output.write(frame[i]);
151             }
152         }finally {
153             if (output!= null) {
154                 System.out.println("Image created");
155                 output.close();
156             }else {
157                 System.out.println("FileOutputStream writer did NOT open - image_error_free NOT found");
158             }
159         }
160
```

Εικόνα με σφάλμα:

- Γίνεται request εικόνας με σφάλμα
- Διατηρούμε τις default ρυθμίσεις για την λήψη της εικόνας
- Ελέγχουμε τα δύο πρώτα και δύο τελευταία bytes της εικόνας, ώστε να καταλάβουμε αν έχει σταλθεί σωστά
- Αποθηκεύουμε την εικόνα ως binary αρχείο jpg

```
164
165         // IMAGE WITH ERROR REQUEST
166         System.out.println("Request image with error package");
167         modem.write(image_with_error.getBytes());
168         ArrayList<Integer> image_error  = new ArrayList<Integer>();
169
170         // Get image with error
171         for (;;) {
172             try {
173                 k = modem.read();
174                 if (k==-1) {
175                     break;
176                 }
177                 image_error.add(k);
178             }catch (Exception x) {
179                 break;
180             }
181         }
182
183         // Image received is stored in array
184         int[] frame_error = new int[image_error.size()];            // store image with error in array
185         for (int i=0 ; i<image_error.size() ; i++) {
186             frame_error[i] = image_error.get(i);
187         }
188
189         // check if image is correct
190         if (frame_error[0]==0xFF && frame_error[1]==0xD8 && frame_error[image_error.size()-2]==0xFF && frame_error[image_error.size()-1]==0xD9
191             System.out.println("Image with error is correct");
192         }else {
193             System.out.println("Image with error is NOT correct  -  Abort mission");
194         }
195
196         // Write image with error in file
197         FileOutputStream output_error = null;
198         try {
199             output_error = new FileOutputStream("image_with_error.jpg");
200             for (int i=0 ; i<image_error.size() ; i++) {
201                 output_error.write(frame_error[i]);
202             }
203         }finally {
204             if (output_error!= null) {
205                 System.out.println("Image with error created");
206                 output_error.close();
207             }else {
208                 System.out.println("FileOutputStream writer did NOT open - image_with_error NOT found");
209             }
210         }
211
```

GPS πακέτα και εικόνα:

- Γινεται request για 90 ίχνη, με αρχική θέση την “0000” από τη διαδρομή X=1
- Ξεκινάμε την αποθήκευση του πακέτου, όταν φτάσει η τιμή “$”
- Η λήψη κάθε πακέτου σταματάει όταν φτάσει η έκφραση “\n”
- Τύπος πακέτου: *$GPGGA,045208.000,4037.6331,N,02257.5633,E,1,07,1.5,57.8,M,36.1,M,,0000*6D*
- Κάνουμε request για λήψη δεδομένων GPS
- Αποθηκεύουμε τις γεωγραφικες συντεταγμένες (γεωγραφικό μήκος και πλάτος)
- Μετατρέπουμε τις τιμες των γεωγραφικών συντεταγμένων σε γωνία, λεπτά και δευτερόλεπτα
- Γίνεται request για λήψη εικόνας GPS
- Επιλέγουμε τα 5 πρώτα ίχνη, που απέχουν τουλάχιστον 10 δευτερόλεπτα μεταξύ τους
- Ελέγχουμε τα δύο πρώτα και δύο τελευταία bytes της εικόνας, ώστε να καταλάβουμε αν έχει σταλθεί σωστά
- Αποθηκεύουμε την εικόνα ως binary αρχείο jpg

```
        // GPS DATA REQUEST
        System.out.println("Request gps data package");
        modem.write(gps.getBytes());
        ArrayList<String> gps_data = new ArrayList<String>();
        String gps_buffer = new String();                       // store gps data
        boolean reader = false;                                 // flag enable/disable reading from modem

        // Get gps data/positions/traces
        for (;;) {
            try {
                k = modem.read();
                if (k==-1) {
                    break;                                      // finished
                }else {
                    if ((char)k=='$') {                         // gps trace starts with '$' and ends with '\n'
                        reader = true;                          // start reading trace
                    }
                    if (reader==true) {
                        if ((char)k=='\n') {                    // reached end of trace
                            reader = false;                     // reset reader
                            gps_data.add(gps_buffer);           // add trace  -  example of gps_data values: $GPGGA,045208.000,4037.6331,
                            gps_buffer = "";                    // reset buffer
                            continue;                           // goto next iteration
                        }
                        gps_buffer = gps_buffer + (char)k;      // store package of trace in buffer
                    }
                }
            }catch (Exception x) {
                break;
            }
        }

        // MAKE GPS IMAGE REQUEST
        String latitude = new String();;                        // north latitude coordinates
        int lat;
        String longitude = new String();;                       // east longitude coordinates
        int lon;
        String code = new String("T=");
        String decimal = new String("");

        // initialize longitude
        longitude = gps_data.get(0).substring(31,35);           // (31,35) = gps angle & gps minutes of longitude
        decimal = gps_data.get(0).substring(36,40);             // (36,40) = gps seconds of longitude
        lon = Integer.parseInt(decimal);                        // convert String "decimal" into Integer
        lon = (int)(lon*0.006);                                 // convert the decimal part of minutes to seconds
        decimal = Integer.toString(lon);                        // convert seconds to String
        longitude = longitude + decimal;                        // longitude = AABBCC  -  AA = angles - BB = minutes - CC = seconds

        // initialize latitude
        latitude = gps_data.get(0).substring(18,22);            // (18,22) = gps angle & gps minutes of latitude
        decimal = gps_data.get(0).substring(23,27);             // (23,27) = gps seconds of latitude
        lat = Integer.parseInt(decimal);                        // convert String "decimal" into Integer
        lat = (int)(lat*0.006);                                 // convert the decimal part of minutes to seconds
        decimal = Integer.toString(lat);                        // convert seconds to String
        latitude = latitude + decimal;                          // latitude = DDEEZZ  -  DD = angles - EE = minutes - ZZ = seconds

        // code: "T=AABBCCDDEEZZ"
        code = code + longitude;
        code = code + latitude;

        // initialize time
        String gps_time = new String();
        gps_time = gps_data.get(0).substring(7,13);             // (7,13) = time

```

```
        // We need at least 4 traces, which are at least 4 seconds apart from each other
        // I choose the first 5 traces, which are 10 seconds apart
        int trace_counter = 0;                                      // 5 traces
        int hour1, hour2, min1, min2, sec1, sec2;
        int time_dif = 0;                                           // time difference in seconds
        String gps_time2 = new String();
        boolean hour_carry, minute_carry;
        for (int i=1 ; i<gps_data.size() ; i++) {
            hour1 = Integer.parseInt(gps_time.substring(0,1));      // (0,1) = hours
            min1 = Integer.parseInt(gps_time.substring(2,3));       // (2,3) = minutes
            sec1 = Integer.parseInt(gps_time.substring(4,6));       // (4,6) = seconds
            gps_time2 = gps_data.get(i).substring(7,13);
            hour2 = Integer.parseInt(gps_time2.substring(0,1));
            min2 = Integer.parseInt(gps_time2.substring(2,3));
            sec2 = Integer.parseInt(gps_time2.substring(4,6));
            time_dif = 0;
            hour_carry = false;
            minute_carry = false;
            if (sec2-sec1>=0) {
                time_dif = sec2-sec1;
            }else {
                minute_carry = true;
                time_dif = sec2-sec1+60;
            }
            if (minute_carry==true) {
                min1++;
            }
            if (min2-min1>=0) {
                time_dif = time_dif + 60*(min2-min1);
            }else {
                hour_carry = true;
                time_dif = time_dif + 60*(min2-min1+60);
            }
            if (hour_carry==true) {
                hour1++;
            }
            time_dif = time_dif + 60*60*(hour2-hour1);
            if (time_dif>=10) {                                     // found a trace
                gps_time = gps_time2;
                trace_counter++;
                if (trace_counter>4) {                              // found 5 traces
                    break;
                }
                decimal = "";
                code = "T=";

                longitude = gps_data.get(i).substring(31,35);
                decimal = gps_data.get(i).substring(36,40);
                lon = Integer.parseInt(decimal);
                lon = (int)(lon*0.006);
                decimal = Integer.toString(lon);
                longitude = longitude + decimal;

                latitude = gps_data.get(i).substring(18,22);
                decimal = gps_data.get(i).substring(23,27);
                lat = Integer.parseInt(decimal);
                lat = (int)(lat*0.006);
                decimal = Integer.toString(lat);
                latitude = latitude + decimal;

                code = code + longitude;
                code = code + latitude;
                gps_image_code = gps_image_code + code;                 // adding parameter T after gps image request code
            }
        }
        gps_image_code = gps_image_code + '\r';

```

```java
        // REQUEST & WRITE IMAGE
        modem.write(gps_image_code.getBytes());                         // request gps image
        ArrayList<Integer> gps_image = new ArrayList<Integer>();
        // Get image from server
        for (;;) {
            try {
                k = modem.read();
                if (k==-1) {
                    break;
                }
                gps_image.add(k);                                       // store image bits
            }catch (Exception x) {
                break;
            }
        }

        // Image received is stored in array
        int[] gps_frame = new int[gps_image.size()];
        for (int i=0 ; i<gps_image.size() ; i++) {
            gps_frame[i] = gps_image.get(i);                            // store gps image in array
        }

        // check if image is correct
        if (gps_frame[0]==0xFF && gps_frame[1]==0xD8 && gps_frame[gps_image.size()-2]==0xFF && gps_frame[gps_image.size()-1]==0xD9) {
            System.out.println("GPS image is correct");
        }else {
            System.out.println("GPS image is NOT correct  -  Abort mission");
        }

        // Write gps image in file
        FileOutputStream output_gps = null;
        try {
            output_gps = new FileOutputStream("GPS_image.jpg");
            for (int i=0 ; i<gps_image.size() ; i++) {
                output_gps.write(gps_frame[i]);
            }
        }finally {
            if (output_gps!= null) {
                System.out.println("GPS image created");
                output_gps.close();
            }else {
                System.out.println("FileOutputStream writer did NOT open - gps_image NOT found");
            }
        }
```

<u>ARQ (ACK-NACK) πακέτα:</u>

- Στέλνουμε συνεχόμενα request για αποστολή πακέτων από την Ithaki
- Η λήψη κάθε πακέτου σταματάει όταν φτάσει η έκφραση “PSTOP”
- Ελέγχουμε αν το πακέτο έφτασε σωστά και αποφασίζουμε αναλόγως αν θα ζητήσουμε επανεκπομπή ή όχι
- Μετράμε το χρόνο που χρειάζεται να γίνει σωστή λήψη κάθε πακέτου
- Μετράμε τον αριθμών των ACK και NACK requests
- Η διαδικασία τερματίζει μόλις περάσουν 5 λεπτά
- Αποθηκεύουμε σε file τους χρόνους απόκρισης κάθε πακέτου, από τη στιγμή του request, μέχρι τη σωστή λήψη τους
- Μετράμε τον αριθμό επανεκπομπών κάθε πακέτου (ο αριθμός εκπομπών ισούται με τον αριθμό επανεκπομπών+1)

```
        // ARQ - ACK & NACK REQUEST
        System.out.println("Request ack & nack package");
        ArrayList<Long> time_ack = new ArrayList<Long>();               // response time until correct send
        ArrayList<Integer> resend_list = new ArrayList<Integer>();      // number of resends for every package
        buffer = "";
        boolean resend = false;                                         // resend a package
        int resend_counter = 0;                                         // number of time a package was resend
        int resend_counter_total = 0;                                   // number of time packages was resend - nack counter
        int correct_package = 0;                                        // number of packages sent correcty - ack counter
        counter = 0;                                                    // number of total packages
        start_of_5_minites = System.currentTimeMillis();
        long now;
        int fcs;                                                        // frame check sequence
        String encrypted = new String("");;                             // 16 char sequence
        char xor;                                                       // XOR

        // Get ack & nack packages
        for (;;) {
            try {
                buffer = "";
                // ack,nack choice
                if (resend==false) {                                    // ack
                    start = System.currentTimeMillis();
                    modem.write(ack.getBytes());
                }else {
                    modem.write(nack.getBytes());                       // nack
                }
                // read package
                for (;;) {
                    k = modem.read();                                   // PSTART DD-MM-YYYY HH-MM-SS PC <XXXXXXXXXXXXXXXX> FCS PSTOP
                    buffer = buffer + (char)k;
                    if (buffer.contains("PSTOP")) {
                        // System.out.println(buffer);
                        counter++;
                        break;
                    }
                }
                encrypted = buffer.substring(31,47);                    // XXXXXXXXXXXXXXXX
                fcs = Integer.parseInt(buffer.substring(49,52));        // FCS
                xor = (char)(encrypted.charAt(0)^encrypted.charAt(1));
                for (int i=2 ; i<encrypted.length() ; i++) {
                    xor = (char)(xor^encrypted.charAt(i));              // successive XOR
                }
                if (fcs==(int)xor) {
                    resend = false;                                     // ack
                    end = System.currentTimeMillis();
                    passed = end - start;
                    time_ack.add(passed);
                    resend_list.add(resend_counter);
                    resend_counter = 0;                                 // reset
                    correct_package++;
                }else {
                    resend = true;                                      // nack
                    resend_counter++;
                    resend_counter_total++;
                }
                now = System.currentTimeMillis();
                threshold = now - start_of_5_minites;
                if (threshold>1000*60*5){                               // 5 minutes in msec for echo packages request duration
                    System.out.println("5 minutes for ack & nack packages request duration have passed");
                    break;
                }
                continue;
            }catch (Exception x) {
                System.out.println("Exception exit: ack & nack");
                break;
            }
        }

        System.out.println("resend_counter_total = " + resend_counter_total);
        System.out.println("correct_package = " + correct_package);
        System.out.println("counter = " + counter);
```

```
        // MEASUREMENTS
        // ARQ times
        try {
            FileWriter writer = new FileWriter("Response_Times_ARQ.txt", true); // file writer
            for (int i=0 ; i<time_ack.size() ; i++) {
                writer.write(Long.toString(time_ack.get(i)));           // write response time in a file
                temp = "\n";                                            // press enter
                writer.write(temp);
            }
            writer.close();
        }catch (IOException e) {
            System.out.println("An error occurred");
            e.printStackTrace();
        }

        // resend counter
        try {
            FileWriter writer = new FileWriter("Resend.txt", true); // file writer
            for (int i=0 ; i<resend_list.size() ; i++) {
                writer.write(Long.toString(resend_list.get(i)));            // write response time in a file
                temp = "\n";                                            // press enter
                writer.write(temp);
            }
            writer.close();
        }catch (IOException e) {
            System.out.println("An error occurred");
            e.printStackTrace();
        }

        // store counters in file
        try {
            FileWriter writer = new FileWriter("ARQ_counters.txt", true);   // file writer
            writer.write(Long.toString(resend_counter_total));
            temp = "\n";                                                // press enter
            writer.write(temp);
            writer.write(Long.toString(correct_package));
            temp = "\n";                                                // press enter
            writer.write(temp);
            writer.write(Long.toString(counter));
            temp = "\n";                                                // press enter
            writer.write(temp);
            writer.close();
        }catch (IOException e) {
            System.out.println("An error occurred");
            e.printStackTrace();
        }
        System.out.println("ARQ package finished");



        modem.close();
        System.out.println("\nBye ithaki!");
    }
}
```

Πηγαίος Κώδικα:

```
/* Dimitris Pappas
 * AEM: 8391
 * */
package source_code_8391;

import java.io.*;
import java.util.*;
import ithakimodem.Modem;

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

		Modem modem  = new Modem();
		modem.open("ithaki");
		modem.setSpeed(80000);
			// 80 Kbps
		modem.setTimeout(2000);


		String temp = new String("");
		String buffer = new String("");
		int k;
```

```
        String echo = new String("E8185\r");
    String image_error_free = new String("M2680\r");
    String image_with_error = new String("G7471\r");
    String gps = new String("P9202R=1000090\r");                    //
R=XPPPPLL  -  X = presaved route (1)  -  PPPP = starting position (0000)  -  LL =
number of traces (90)
    String gps_image_code = new String("P9202");                    //
follows code T=AABBCCDDEEZZ\r
    String ack = new String("Q7721\r");
    String nack = new String("R3491\r");



        // Log In
        System.out.println("Hello ithaki!\n");
        for (;;){
            try {
                k = modem.read();
                if (k==-1){
                    break;
                }if(k!=-1){
                    // do nothing
                }
                // System.out.print((char)k);
            }catch (Exception x){
```

```
                        break;
                    }
                }

                // ------------------------------------------------------------------------------------
------



                // ECHO PACKAGE REQUEST   -   REMAKE
                System.out.println("Request echo package");
                int counter = 1;
                // number of packages
                long start = System.currentTimeMillis();
            // keep the current time in msec when requesting a package
                long start_of_5_minites = start;
            // keep the current time in msec when writing to modem for the first time
                long end;
                    // keep the current time in msec when package arrived
                long passed;
                // time passed until package received
                long threshold;
                    // time passed since first request
                ArrayList<String> pack = new ArrayList<String>();                    //
store packages
                ArrayList<Long> time = new ArrayList<Long>();
            // store response time
```

```
        // Get echo packages
        for (;;) {
                try{
                        buffer = "";
                        modem.write(echo.getBytes());
// request packages from server
                        start = System.currentTimeMillis();

                        for (;;) {
                                k = modem.read();
        // PSTART DD-MM-YYYY HH-MM-SS PC PSTOP
                                buffer = buffer + (char)(k);
                                if (buffer.contains("PSTOP")) {
// package received
                                        end = System.currentTimeMillis();
                                        passed = end - start;
                                        pack.add(buffer);
                                        time.add(passed);
                                        // System.out.println(buffer);
                                        break;
                                }
                        }

                        threshold = end - start_of_5_minites;
                        counter++;
```

```
                        if (threshold>1000*60*5){
                    // 5 minutes in msec for echo packages request duration
                            System.out.println("5 minutes for echo packages request duration have passed");
                            break;
                        }
                        continue;
                    }catch (Exception x) {
                        break;
                    }
                }

                System.out.println("number of echo packages = " + counter);

                try {
                    FileWriter writer = new FileWriter("Response_Times.txt", true); // file writer
                    for (int i=0 ; i<time.size() ; i++) {
                        writer.write(Long.toString(time.get(i)));
            // write response time in a file
                        temp = "\n";
                // press enter
                        writer.write(temp);
                    }
                    writer.close();
                }catch (IOException e) {
```

```
                System.out.println("An error occurred");
                e.printStackTrace();
            }
            System.out.println("Echo package finished");



            // ------------------------------------------------------------------------------------
------



            // IMAGE ERROR FREE REQUEST
            System.out.println("Request image error free package");
            modem.write(image_error_free.getBytes());
        // request image
            ArrayList<Integer> image = new ArrayList<Integer>();              //
Store image


            // Get image
            for (;;) {
                try {
                    k = modem.read();
                    if (k==-1) {
                        // finished reading
                        break;
                    }
```

```
                    image.add(k);
                // store package in image list
                }catch (Exception x) {
                    break;
                }
            }

            // Image received is stored in array
            int[] frame = new int[image.size()];
        // store image in array
            for (int i=0 ; i<image.size() ; i++) {
                frame[i] = image.get(i);
            }

            // check if image is correct
            if (frame[0]==0xFF && frame[1]==0xD8 && frame[image.size()-
2]==0xFF && frame[image.size()-1]==0xD9) {
                System.out.println("Image is correct");
            }else {
                System.out.println("Image is NOT correct  -  Abort mission");
            }

            // Write image in file
            FileOutputStream output = null;
            // FileOutputStream is meant for writing streams of raw bytes such
as image data to a file
```

```
		try {
			output = new FileOutputStream("image_error_free.jpg");
			for (int i=0 ; i<image.size() ; i++) {
				output.write(frame[i]);
			}
		}finally {
			if (output!= null) {
				System.out.println("Image created");
				output.close();
			}else {
				System.out.println("FileOutputStream writer did NOT open - image_error_free NOT found");
			}
		}


		// ------------------------------------------------------------------------------------------


		// IMAGE WITH ERROR REQUEST
		System.out.println("Request image with error package");
		modem.write(image_with_error.getBytes());
		ArrayList<Integer> image_error  = new ArrayList<Integer>();
```

```
// Get image with error
for (;;) {
    try {
        k = modem.read();
        if (k==-1) {
            break;
        }
        image_error.add(k);
    }catch (Exception x) {
        break;
    }
}

// Image received is stored in array
int[] frame_error = new int[image_error.size()];                    // store image with error in array
for (int i=0 ; i<image_error.size() ; i++) {
    frame_error[i] = image_error.get(i);
}

// check if image is correct
if (frame_error[0]==0xFF && frame_error[1]==0xD8 && frame_error[image_error.size()-2]==0xFF && frame_error[image_error.size()-1]==0xD9) {
        System.out.println("Image with error is correct");
```

```
		}else {
			System.out.println("Image with error is NOT correct  -  Abort mission");
		}

		// Write image with error in file
		FileOutputStream output_error = null;
		try {
			output_error = new FileOutputStream("image_with_error.jpg");
			for (int i=0 ; i<image_error.size() ; i++) {
				output_error.write(frame_error[i]);
			}
		}finally {
			if (output_error!= null) {
				System.out.println("Image with error created");
				output_error.close();
			}else {
				System.out.println("FileOutputStream writer did NOT open - image_with_error NOT found");
			}
		}

		// ------------------------------------------------------------------------------------
```

```
        // GPS DATA REQUEST
        System.out.println("Request gps data package");
        modem.write(gps.getBytes());
        ArrayList<String> gps_data = new ArrayList<String>();
        String gps_buffer = new String();
// store gps data
        boolean reader = false;
// flag enable/disable reading from modem


        // Get gps data/positions/traces
        for (;;) {
                try {
                        k = modem.read();
                        if (k==-1) {
                                break;
                        // finished
                        }else {
                                if ((char)k=='$') {
        // gps trace starts with '$' and ends with '\n'
                                        reader = true;
                // start reading trace
                                }
                                if (reader==true) {
```

```
                                        if ((char)k=='\n') {
        // reached end of trace
                                                reader = false;
                        // reset reader
                                                gps_data.add(gps_buffer);
                    // add trace  -  example of gps_data values:
$GPGGA,045208.000,4037.6331,N,02257.5633,E,1,07,1.5,57.8,M,36.1,M,,0000*6
D
                                                gps_buffer = "";
                    // reset buffer
                                                continue;
                        // goto next iteration
                                        }
                                        gps_buffer = gps_buffer + (char)k;
        // store package of trace in buffer
                                    }
                                }
                        }catch (Exception x) {
                            break;
                        }
                }


                // MAKE GPS IMAGE REQUEST
                String latitude = new String();;
        // north latitude coordinates
                int lat;
```

```
        String longitude = new String();;
    // east longitude coordinates

        int lon;

        String code = new String("T=");

        String decimal = new String("");


        // initialize longitude

        longitude = gps_data.get(0).substring(31,35);
    // (31,35) = gps angle & gps minutes of longitude

        decimal = gps_data.get(0).substring(36,40);
    // (36,40) = gps seconds of longitude

        lon = Integer.parseInt(decimal);
    // convert String "decimal" into Integer

        lon = (int)(lon*0.006);
        // convert the decimal part of minutes to seconds

        decimal = Integer.toString(lon);
    // convert seconds to String

        longitude = longitude + decimal;
    // longitude = AABBCC  -  AA = angles - BB = minutes - CC = seconds


        // initialize latitude

        latitude = gps_data.get(0).substring(18,22);                    //
(18,22) = gps angle & gps minutes of latitude

        decimal = gps_data.get(0).substring(23,27);
    // (23,27) = gps seconds of latitude

        lat = Integer.parseInt(decimal);
    // convert String "decimal" into Integer
```

```
            lat = (int)(lat*0.006);
            // convert the decimal part of minutes to seconds

            decimal = Integer.toString(lat);
        // convert seconds to String

            latitude = latitude + decimal;
        // latitude = DDEEZZ  -  DD = angles - EE = minutes - ZZ = seconds


            // code: "T=AABBCCDDEEZZ"

            code = code + longitude;

            code = code + latitude;



            // initialize time

            String gps_time = new String();

            gps_time = gps_data.get(0).substring(7,13);
        // (7,13) = time


            // We need at least 4 traces, which are at least 4 seconds apart from
each other

            // I choose the first 5 traces, which are 10 seconds apart

            int trace_counter = 0;
            // 5 traces

            int hour1, hour2, min1, min2, sec1, sec2;

            int time_dif = 0;
            // time difference in seconds

            String gps_time2 = new String();

            boolean hour_carry, minute_carry;
```

```
            for (int i=1 ; i<gps_data.size() ; i++) {
                    hour1 = Integer.parseInt(gps_time.substring(0,1));          //
(0,1) = hours
                    min1 = Integer.parseInt(gps_time.substring(2,3));           //
(2,3) = minutes
                    sec1 = Integer.parseInt(gps_time.substring(4,6));           //
(4,6) = seconds
                    gps_time2 = gps_data.get(i).substring(7,13);
                    hour2 = Integer.parseInt(gps_time2.substring(0,1));
                    min2 = Integer.parseInt(gps_time2.substring(2,3));
                    sec2 = Integer.parseInt(gps_time2.substring(4,6));
                    time_dif = 0;
                    hour_carry = false;
                    minute_carry = false;
                    if (sec2-sec1>=0) {
                            time_dif = sec2-sec1;
                    }else {
                            minute_carry = true;
                            time_dif = sec2-sec1+60;
                    }
                    if (minute_carry==true) {
                            min1++;
                    }
                    if (min2-min1>=0) {
                            time_dif = time_dif + 60*(min2-min1);
```

```
        }else {
                hour_carry = true;
                time_dif = time_dif + 60*(min2-min1+60);
        }
        if (hour_carry==true) {
                hour1++;
        }
        time_dif = time_dif + 60*60*(hour2-hour1);
        if (time_dif>=10) {
// found a trace
                gps_time = gps_time2;
                trace_counter++;
                if (trace_counter>4) {
// found 5 traces
                        break;
                }
                decimal = "";
                code = "T=";

                longitude = gps_data.get(i).substring(31,35);
                decimal = gps_data.get(i).substring(36,40);
                lon = Integer.parseInt(decimal);
                lon = (int)(lon*0.006);
                decimal = Integer.toString(lon);
                longitude = longitude + decimal;
```

```
                latitude = gps_data.get(i).substring(18,22);
                decimal = gps_data.get(i).substring(23,27);
                lat = Integer.parseInt(decimal);
                lat = (int)(lat*0.006);
                decimal = Integer.toString(lat);

                latitude = latitude + decimal;


                code = code + longitude;
                code = code + latitude;
                gps_image_code = gps_image_code + code;
// adding parameter T after gps image request code
        }
}
gps_image_code = gps_image_code + '\r';

// REQUEST & WRITE IMAGE
modem.write(gps_image_code.getBytes());
// request gps image
ArrayList<Integer> gps_image = new ArrayList<Integer>();
// Get image from server
for (;;) {
        try {
                k = modem.read();
```

```
                        if (k==-1) {
                                break;
                        }
                        gps_image.add(k);
                // store image bits
                }catch (Exception x) {
                        break;
                }
        }

        // Image received is stored in array
        int[] gps_frame = new int[gps_image.size()];
        for (int i=0 ; i<gps_image.size() ; i++) {
                gps_frame[i] = gps_image.get(i);
        // store gps image in array
        }

        // check if image is correct
        if (gps_frame[0]==0xFF && gps_frame[1]==0xD8 &&
gps_frame[gps_image.size()-2]==0xFF && gps_frame[gps_image.size()-1]==0xD9)
{
                System.out.println("GPS image is correct");
        }else {
                System.out.println("GPS image is NOT correct  -  Abort
mission");
        }
```

```
		// Write gps image in file
		FileOutputStream output_gps = null;
		try {
			output_gps = new FileOutputStream("GPS_image.jpg");
			for (int i=0 ; i<gps_image.size() ; i++) {
				output_gps.write(gps_frame[i]);
			}
		}finally {
			if (output_gps!= null) {
				System.out.println("GPS image created");
				output_gps.close();
			}else {
				System.out.println("FileOutputStream writer did NOT
open - gps_image NOT found");
			}
		}



		// ----------------------------------------------------------------------------------
------



		// ARQ - ACK & NACK REQUEST
		System.out.println("Request ack & nack package");
```

```
        ArrayList<Long> time_ack = new ArrayList<Long>();
// response time until correct send

        ArrayList<Integer> resend_list = new ArrayList<Integer>();
// number of resends for every package

        buffer = "";

        boolean resend = false;
                // resend a package


        int resend_counter = 0;
                // number of time a package was resend

        int resend_counter_total = 0;
        // number of time packages was resend - nack counter

        int correct_package = 0;
        // number of packages sent correcty - ack counter

        counter = 0;
                // number of total packages

        start_of_5_minites = System.currentTimeMillis();

        long now;

        int fcs;
                        // frame check sequence

        String encrypted = new String("");;
        // 16 char sequence

        char xor;
                        // XOR


        // Get ack & nack packages

        for (;;) {

                try {
```

```
                        buffer = "";
                        // ack,nack choice
                        if (resend==false) {
                // ack
                                start = System.currentTimeMillis();
                                modem.write(ack.getBytes());
                        }else {
                                modem.write(nack.getBytes());
                // nack
                        }
                        // read package
                        for (;;) {
                                k = modem.read();
                        // PSTART DD-MM-YYYY HH-MM-SS PC <XXXXXXXXXXXXXXXX>
FCS PSTOP
                                buffer = buffer + (char)k;
                                if (buffer.contains("PSTOP")) {
                                        // System.out.println(buffer);
                                        counter++;
                                        break;
                                }
                        }
                        encrypted = buffer.substring(31,47);
        // XXXXXXXXXXXXXXXX
                        fcs = Integer.parseInt(buffer.substring(49,52));
        // FCS
```

```
                    xor = (char)(encrypted.charAt(0)^encrypted.charAt(1));
                    for (int i=2 ; i<encrypted.length() ; i++) {
                        xor = (char)(xor^encrypted.charAt(i));
// successive XOR
                    }
                    if (fcs==(int)xor) {

                        resend = false;
                    // ack
                        end = System.currentTimeMillis();
                        passed = end - start;
                        time_ack.add(passed);
                        resend_list.add(resend_counter);
                        resend_counter = 0;
                    // reset
                        correct_package++;
                    }else {
                        resend = true;
                    // nack
                        resend_counter++;
                        resend_counter_total++;
                    }
                    now = System.currentTimeMillis();
                    threshold = now - start_of_5_minites;
                    if (threshold>1000*60*5){
                // 5 minutes in msec for echo packages request duration
```

```
                                System.out.println("5 minutes for ack & nack
packages request duration have passed");
                                break;
                        }
                        continue;
                }catch (Exception x) {
                        System.out.println("Exception exit: ack & nack");
                        break;
                }
        }

        System.out.println("resend_counter_total = " +
resend_counter_total);
        System.out.println("correct_package = " + correct_package);
        System.out.println("counter = " + counter);

        // MEASUREMENTS
        // ARQ times
        try {
                FileWriter writer = new FileWriter("Response_Times_ARQ.txt",
true); // file writer
                for (int i=0 ; i<time_ack.size() ; i++) {
                        writer.write(Long.toString(time_ack.get(i)));
    // write response time in a file
                        temp = "\n";
                // press enter
```

```
                    writer.write(temp);
                }
                writer.close();
            }catch (IOException e) {
                System.out.println("An error occurred");
                e.printStackTrace();
            }

            // resend counter
            try {
                FileWriter writer = new FileWriter("Resend.txt", true);     // file
writer
                for (int i=0 ; i<resend_list.size() ; i++) {
                    writer.write(Long.toString(resend_list.get(i)));
        // write response time in a file
                    temp = "\n";
                // press enter
                    writer.write(temp);
                }
                writer.close();
            }catch (IOException e) {
                System.out.println("An error occurred");
                e.printStackTrace();
            }
```

```
        // store counters in file
        try {
                FileWriter writer = new FileWriter("ARQ_counters.txt", true);
// file writer
                writer.write(Long.toString(resend_counter_total));

                temp = "\n";
        // press enter
                writer.write(temp);
                writer.write(Long.toString(correct_package));
                temp = "\n";
        // press enter
                writer.write(temp);
                writer.write(Long.toString(counter));
                temp = "\n";
        // press enter
                writer.write(temp);
                writer.close();
        }catch (IOException e) {
                System.out.println("An error occurred");
                e.printStackTrace();
        }
        System.out.println("ARQ package finished");
```

```
                modem.close();
                System.out.println("\nBye ithaki!");
        }
}
```