```java
/*
    THEODOROS LIOUPIS  AEM:09733
    Experimental Virtual Lab - Java Serial Communications Programming
    Computer Networks I, 6th Semester, ECE AUTh
*/

//import ithakimodem.*;
import java.io.*;

public class virtualModem {
    public static void main(String[] param){
        Console console = System.console();
        virtualModem virtualModem = new virtualModem();
        Modem modem;
        modem = new Modem();
        modem.setSpeed(80000);
        modem.setTimeout(500);
        modem.open("ithaki");
        virtualModem.message_request(modem);
        while(true){
            System.out.println("\nEnter the XXXXX_request_code: (Type EXIT to terminate the modem)");
            String requestCode = console.readLine() + "\r";
            if(requestCode.indexOf("E") == 0 && requestCode.length() == 6){
                virtualModem.echoPacketLoop(modem, requestCode);
                //modem.write(requestCode.getBytes());
                //virtualModem.message_request(modem);
                continue;
            }
            else if(requestCode.indexOf("M") == 0 && requestCode.length() == 6){
                virtualModem.image_request(modem, requestCode, "error_free_image");
                continue;
            }
            else if(requestCode.indexOf("G") == 0 && requestCode.length() == 6){
                virtualModem.image_request(modem, requestCode, "error_image");
                continue;
            }
            else if(requestCode.indexOf("P") == 0 && (requestCode.length() == 6 || requestCode.length() == 15)){
                virtualModem.gps_request(modem, requestCode);
                continue;
            }
            else if(requestCode.indexOf("P") == 0 && requestCode.length() > 18){
                virtualModem.image_request(modem, requestCode, "gps_image");
                continue;
            }
            else if((requestCode.indexOf("Q") == 0 ||  requestCode.indexOf("R") == 0) & requestCode.length() == 6){
                String ackRequestCode = "";
                String nackRequestCode = "";
                if(requestCode.indexOf("R") == 0){
                    System.out.println("You entered a NACK request code\nPlease enter the ACK request code: ");
                    ackRequestCode = console.readLine() + "\r";
                    nackRequestCode = requestCode;
                }
                else if(requestCode.indexOf("Q") == 0){
                    System.out.println("You entered an ACK request code\nPlease enter the NACK request code as well: ");
                    ackRequestCode = requestCode;
```

```
                    nackRequestCode = console.readLine() + "\r";
                }
                else{
                    continue;
                }
                virtualModem.ARQ(modem, ackRequestCode, nackRequestCode);
                System.out.println("\nARQ process finished");
                continue;
            }
            else if(requestCode.indexOf("exit") > -1 || requestCode.indexOf("Exit") > -1 || requestCode.indexOf("EXIT") > -1){
                modem.close();
                break;
            }
            else{
                System.out.println("\nInvalid Request Code");
                continue;
            }
        }
        modem.close();
    }

    public static void message_request(Modem modem){
        int inByte;
        String rxMessage = "";
        while(true) {
            try {
                inByte = modem.read();
                if (inByte == -1){
                    System.out.println("\nConnection lost");
                    break;
                }
                System.out.print((char)inByte);
                rxMessage += (char)inByte;
                if (rxMessage.indexOf("\r\n\n\n")>-1){
                    System.out.println("Welcome Message ended");
                    break;
                }
                if (rxMessage.indexOf("PSTOP")>-1){
                    System.out.println("\nMessage ended");
                    System.out.println("EchoPacket received");
                    break;
                }
            }
            catch (Exception x) {
                break;
            }
        }
    }

    public void echoPacketLoop(Modem modem, String requestCode){
        long loopStartTime = System.nanoTime();
        int counter = 1;
        while((System.nanoTime() - loopStartTime)/1000000 < 300000){
            modem.write(requestCode.getBytes());
            long packetStartTime = System.nanoTime();
            virtualModem.message_request(modem);
            long packetEndTime = System.nanoTime();
            long responseTime = packetEndTime - packetStartTime;
```

```
114             System.out.println("Response time: " + ((packetEndTime -
    packetStartTime)/1000000));
115             try {
116                 FileWriter outputFile = new
    FileWriter("echoPacket_response_time.txt", true);
117                 outputFile.write(String.valueOf(counter));
118                 outputFile.write("\t");
119                 outputFile.write(String.valueOf(responseTime/1000000));
120                 outputFile.write("\n");
121                 outputFile.close();
122                 counter++;
123             } catch (IOException e) {
124                 e.printStackTrace();
125             }
126         }
127     }
128 
129     public void image_request(Modem modem, String requestCode, String imageName){
130         int inByte;
131         String rxMessage = "";
132         if(modem.write(requestCode.getBytes())){
133             System.out.println("Image request received by the Server");
134         }
135         try {
136             FileOutputStream outputfile = new FileOutputStream(imageName + ".jpeg");
137             while(true) {
138                 inByte = modem.read();
139                 if (inByte == -1){
140                     System.out.println("\nConnection lost");
141                     break;
142                 }
143                 rxMessage += inByte;
144                 outputfile.write((byte) inByte);
145                 /*
146                 if (rxMessage.indexOf("255216")>-1){
147                     System.out.print(" Found Start Delimiter ");
148                 }*/
149                 if (rxMessage.indexOf("255217")>-1) {
150                     //System.out.println(" Found End Delimiter");
151                     System.out.println("Image Created");
152                     if(requestCode.indexOf("M") == 0){
153                         System.out.println("Image with no errors received");
154                     }
155                     else if(requestCode.indexOf("G") == 0){
156                         System.out.println("Image with errors received");
157                     }
158                     else{
159                         System.out.println("GPS image received");
160                     }
161                     break;
162                 }
163             }
164         }
165         catch (Exception e) {
166             System.out.println("\nError receiving image");
167         }
168     }
169 
170     public void gps_request(Modem modem, String requestCode){
```

```
171         int inByte;
172         String rxMessage = "";
173         if(modem.write(requestCode.getBytes())){
174             System.out.println("\nGPS request received by the Server\n");
175         }
176         try{
177             while(true) {
178                 inByte = modem.read();
179                 if (inByte == -1){
180                     System.out.println("\nConnection lost");
181                     break;
182                 }
183                 System.out.print((char)inByte);
184                 rxMessage += (char)inByte;
185                 if (rxMessage.indexOf("STOP ITHAKI GPS TRACKING\r\n") >-1 ){
186                     System.out.println("\nGPS message ended");
187                     break;
188                 }
189             }
190         }
191         catch (Exception e){
192             System.out.println("\nError receiving GPS signal");
193         }
194     }
195 
196     public void ARQ(Modem modem, String ackRequestCode, String nackRequestCode){
197         long arqStartTime = System.nanoTime();
198         String requestCode = ackRequestCode;
199         int numOfRepeats = 0;
200         int packetCounter = 1;
201         int sumOfAcks = 1;
202         int sumOfNacks = 0;
203         int[] sumOfRepeats = new int[15];
204         for(int i = 0 ; i < 10 ; i++){
205             sumOfRepeats[i] = 0;
206         }
207         long packetStartTime = System.nanoTime();
208         while (true){
209             int inByte;
210             String rxMessage = "";
211             String infoPacket = "";
212             String fcsTransmitted = "";
213             int fcsCounter = 3;
214             modem.write(requestCode.getBytes());
215             while(true) {
216                 try {
217                     inByte = modem.read();
218                     if (inByte == -1){
219                         System.out.println("\nConnection lost");
220                         break;
221                     }
222                     System.out.print((char)inByte);
223                     rxMessage += (char)inByte;
224                     if(rxMessage.indexOf("<") > -1 && rxMessage.indexOf(">") < 0 &&
((int) inByte != 60)){
225                         infoPacket += (char)inByte;
226                     }
227                     if(rxMessage.indexOf("<") > -1 && rxMessage.indexOf(">") > -1 &&
((int) inByte != 32) && ((int) inByte != 62) && (fcsCounter > 0)){
228                         fcsTransmitted += (char)inByte;
```

```
                        fcsCounter--;
                    }
                    if (rxMessage.indexOf("PSTOP")>-1){
                        System.out.println("\nARQ packet received");
                        break;
                    }
                }
                catch (Exception x) {
                    break;
                }
            }
            System.out.println(infoPacket);
            int integerFcsTransmitted = Integer.parseInt(fcsTransmitted);
            System.out.println("The transmitted FCS is: " + integerFcsTransmitted);
            int fcsReceived = infoPacket.charAt(0);
            for(int i = 1 ; i < 16 ; i++){
                fcsReceived^=infoPacket.charAt(i);
            }
            System.out.println("The received FCS is: " + fcsReceived);
            if (integerFcsTransmitted != fcsReceived){
                System.out.println("PACKET WITH ERROR!!");
                numOfRepeats++;
                requestCode = nackRequestCode;
                sumOfNacks++;
                continue;
            }
            else if ((System.nanoTime() - arqStartTime)/1000000 < 240000){
                long packetEndTime = System.nanoTime();
                long responseTime = packetEndTime - packetStartTime;
                System.out.println("Error-free packet!!");
                System.out.println("Repeat times: " + numOfRepeats + "\n ");
                try {
                    FileWriter outputFile = new FileWriter("ARQ_response_time.txt",
true);
                    outputFile.write(String.valueOf(packetCounter));
                    outputFile.write("\t");
                    outputFile.write(String.valueOf(numOfRepeats));
                    outputFile.write("\t");
                    outputFile.write(String.valueOf(responseTime/1000000));
                    outputFile.write("\n");
                    outputFile.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
                packetCounter++;
                if(numOfRepeats < 15){
                    sumOfRepeats[numOfRepeats]++;
                }
                numOfRepeats = 0;
                requestCode = ackRequestCode;
                sumOfAcks++;
                packetStartTime = System.nanoTime();
                continue;
            }
            else{
                sumOfAcks++;
                modem.write(ackRequestCode.getBytes());
                break;
            }
        }
```

```
288         System.out.println("Sum of ACKS: " + sumOfAcks);
289         System.out.println("Sum of NACKS: " + sumOfNacks);
290         try {
291             FileWriter outputFile = new
    FileWriter("ARQ_probability_distribution.txt", true);
292             outputFile.write(String.valueOf("Sum of ACKS: " + sumOfAcks));
293             outputFile.write("\n");
294             outputFile.write(String.valueOf("Sum of NACKS: " + sumOfNacks));
295             outputFile.write("\n");
296             for(int i = 0 ; i < 10 ; i++){
297                 System.out.println("Sum of " + i + " repeats: " + sumOfRepeats[i]);
298                 outputFile.write(String.valueOf(i));
299                 outputFile.write("\t");
300                 outputFile.write(String.valueOf(sumOfRepeats[i]));
301                 outputFile.write("\n");
302             }
303             outputFile.close();
304         } catch (IOException e) {
305             e.printStackTrace();
306         }
307     }
308 }
```