Source code

Δίκτυα Υπολογιστών Ι

Ρόντος Έκτορας – Θωμάς

ΑΕΜ: 9477

Ο κώδικας χωρίζεται σε 2 αρχεία. Το VirtualModem.java όπου αναπτύχθηκαν όλες οι μέθοδοι για τις ζητούμενες λειτουργείες και το UserApplication.java όπου βρίσκεται η main του προγράμματος.

## VirtualModem.java

import java.io.\*;  
import java.time.LocalDateTime;  
import java.time.format.DateTimeFormatter;  
import java.util.ArrayList;  
  
import ithakimodem.\*;  
  
public class VirtualModem {  
 public final String DATETIME\_PATTERN = "yyyy-MM-dd-kk-mm-ss-S";  
  
 private Modem modem;  
 private String log;  
  
 public VirtualModem(int speed, int timeout){  
 modem = new Modem(speed);  
  
 modem.setTimeout(timeout);  
 modem.open("ithaki");  
  
 log = "-------------------------[" + LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN)) + "]-------------------------\n";  
 }  
  
 */\*\*  
 \* Method read  
 \* <br><br>  
 \* Output incoming bytes and log them.  
 \* Stop on given sequence.  
 \*  
 \** ***@param*** *stopOn String sequence to stop reading  
 \*/* public void read(String stopOn){  
 int k;  
  
 for (;;) {  
 try {  
 k = modem.read();  
 } catch (Exception e) {  
 System.*out*.println(e);  
 break;  
 }  
  
 if (k == -1){  
 System.*out*.println("\nConnection error\nReturned "+k);  
 break;  
 }  
  
 addToLog((char)k);  
 System.*out*.print((char)k);  
  
 if(getLog().endsWith(stopOn)){  
 System.*out*.println("\n---STOP---");  
 break;  
 }  
 }  
 }  
  
 */\*\*  
 \* Method write  
 \*  
 \** ***@param*** *string String to write  
 \*  
 \** ***@return*** *True or false  
 \*/* public boolean write(String string){  
 try{  
 if(modem.write(string.getBytes())){  
 addToLog("\n-Write:"+string+"\n");  
 return true;  
 } else{  
 System.*out*.println("\nConnection Error\nUnable to write");  
 return false;  
 }  
 } catch(Exception e){  
 System.*out*.println(e);  
 return false;  
 }  
 }  
  
 */\*\*  
 \* Method echoPacketTest  
 \* <br><br>  
 \* Test system response time with echo packets.  
 \*  
 \** ***@param*** *code The request code  
 \** ***@param*** *duration Duration of testing in milliseconds  
 \*/* public void echoPacketTest(String code, long duration){  
 FileOutputStream file = null;  
 int k;  
 long start, send, delay;  
 String echoPacket;  
  
 try {  
 file = new FileOutputStream("data/G1\_"+LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+".csv");  
  
 addToLog("\nStarting echo packet test ("+  
 LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+  
 ")\n"+  
 "\n -Write:"+code+"\n");  
 start = System.*currentTimeMillis*();  
 for (;;) {  
 echoPacket = "";  
  
 if (!modem.write(code.getBytes())) {  
 System.*out*.println("\nConnection Error\nUnable to write");  
 break;  
 }  
 send = System.*currentTimeMillis*();  
  
 for (;;) {  
 try {  
 k = modem.read();  
 } catch (Exception e) {  
 System.*out*.println(e);  
 break;  
 }  
  
 if (k == -1) {  
 System.*out*.println("\nConnection error\nReturned " + k);  
 break;  
 }  
  
 echoPacket += (char) k;  
  
 if (echoPacket.endsWith("PSTOP")) {  
 delay = System.*currentTimeMillis*() - send;  
 file.write((delay+"\n").getBytes());  
 System.*out*.println(echoPacket + " " + delay + "ms");  
 break;  
 }  
 }  
  
 if (System.*currentTimeMillis*() - start >= duration) {  
 addToLog("\nFinished echo packet test ("+LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+")\n");  
 System.*out*.println("\n---STOP---");  
 break;  
 }  
 }  
 } catch(Exception e){  
 System.*out*.println(e);  
 } finally{  
 if(file!=null){  
 try {  
 file.close();  
 } catch (Exception e){  
 System.*out*.println(e);  
 }  
 }  
 }  
 }  
  
 */\*\*  
 \* Method saveImage  
 \* <br><br>  
 \* Download and save an image.  
 \*  
 \** ***@param*** *imageId String identifying the image (E1, E2, M1)  
 \*/* public void saveImage(String imageId){  
 int k;  
 ArrayList<Integer> imageContent = new ArrayList<>();  
 FileOutputStream imageFile = null;  
  
 try{  
 addToLog("\nStart downloading image "+imageId+" ("+LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+")\n");  
 System.*out*.println("Start downloading image...");  
 for (;;) {  
 try {  
 k = modem.read();  
 } catch(Exception e){  
 System.*out*.println(e);  
 break;  
 }  
  
 if (k==-1){  
 System.*out*.println("\nConnection error\nReturned "+k);  
 break;  
 }  
  
 imageContent.add(k);  
  
 if(imageContent.size() >=2 && imageContent.get(imageContent.size()-2) == 255 && imageContent.get(imageContent.size()-1) == 217){  
 break;  
 }  
 }  
 addToLog("\nFinished downloading image "+imageId+" ("+LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+")\n");  
 System.*out*.println("Finished downloading image");  
  
 addToLog("\nStart creating image "+imageId+" ("+LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+")\n");  
 System.*out*.println("Start creating image...");  
  
 imageFile = new FileOutputStream("img/"+imageId+"\_"+LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+".jpg");  
  
 for(int i=0; i<imageContent.size(); i++){  
 if(imageContent.get(i) == 255 && imageContent.get(i+1) == 216){ //Look for starting point  
 for(int j=i; j<imageContent.size(); j++){  
 imageFile.write(imageContent.get(j));  
 }  
 break;  
 }  
 }  
  
 addToLog("\nFinished creating image "+imageId+" ("+LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+")\n");  
 System.*out*.println("Finished creating image");  
  
 } catch(Exception e){  
 System.*out*.println(e);  
 } finally{  
 if(imageFile != null){  
 try {  
 imageFile.close();  
 } catch (Exception e){  
 System.*out*.println(e);  
 }  
 }  
 }  
  
 }  
  
 */\*\*  
 \* Method getGpsTraces  
 \* <br><br>  
 \* Read returned traces from Ithaki, choose specific ones based on timeBetweenTraces  
 \* and return code with coordinates.  
 \*  
 \** ***@param*** *timeBetweenTraces Time in seconds between returned traces.  
 \*  
 \** ***@return*** *Sting Follow up code with traces' coordinates (Τ=ΑΑΒΒΓΓΔΔΕΕΖΖΤ=...)  
 \*/* public String getGpsTraces(int timeBetweenTraces){  
 int k, index;  
 String currentTrace = "", result = "";  
 ArrayList<String> traces= new ArrayList<>();  
  
 addToLog("\nGet GPS traces ("+  
 LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+")\n");  
  
 for (;;) {  
 try {  
 k = modem.read();  
 } catch (Exception e) {  
 System.*out*.println(e);  
 break;  
 }  
  
 if (k == -1){  
 System.*out*.println("\nConnection error\nReturned "+k);  
 break;  
 } else if ((char)k == '$'){  
 currentTrace = "";  
 }  
  
 currentTrace += (char)k;  
  
 if(currentTrace.startsWith("$GPGGA") && currentTrace.endsWith("\n")){ // Keep trace in ArrayList  
 traces.add(currentTrace);  
 } else if(currentTrace.endsWith("STOP ITHAKI GPS TRACKING")){ //Stop reading  
 break;  
 }  
 }  
  
 index = 0;  
 while (index<traces.size()){  
 addToLog(traces.get(index)); // Log chosen traces  
 System.*out*.println(traces.get(index));  
 result += "T=" + traces.get(index).substring(31, 35) +  
 (int)Math.*round*( Integer.*parseInt*(traces.get(index).substring(36, 40) ) \* 0.006) +  
 traces.get(index).substring(18, 22) +  
 (int)Math.*round*( Integer.*parseInt*(traces.get(index).substring(23, 27) ) \* 0.006);  
  
 index += timeBetweenTraces;  
 }  
  
 return result;  
 }  
  
 */\*\*  
 \* Method arqTest  
 \* <br><br>  
 \* Test system response time with ARQ mechanism.  
 \*  
 \** ***@param*** *ack ACK result code  
 \** ***@param*** *nack NACK result code  
 \** ***@param*** *duration Duration of testing in milliseconds  
 \*/* public void arqTest(String ack, String nack, long duration){  
 FileOutputStream responseFile = null, repeatsFile = null;  
 int k, receivedFCS, calculatedFCS, acksSent = 0, nacksSent = 0, repeats = 0;  
 long start, send = 0, delay;  
 String response, code = ack;  
 byte[] message;  
  
 try {  
 responseFile = new FileOutputStream("data/G2\_"+  
 LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+".csv");  
 repeatsFile = new FileOutputStream("data/G3\_"+  
 LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+".csv");  
  
 addToLog("\nStarting ARQ test ("+  
 LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+  
 ")\n"+  
 "\n -ACK: "+ack+  
 "\n -NACK: "+nack+"\n");  
 start = System.*currentTimeMillis*();  
 for (;;) { //Code sending loop  
 response = "";  
  
 if (!modem.write(code.getBytes())) {  
 System.*out*.println("\nConnection Error\nUnable to write");  
 break;  
 }  
  
 if(code.equals(ack)) {  
 send = System.*currentTimeMillis*();  
 acksSent ++;  
 } else{  
 nacksSent ++;  
 repeats ++;  
 }  
  
 for (;;) { //Reading loop  
 try {  
 k = modem.read();  
 } catch (Exception e) {  
 System.*out*.println(e);  
 break;  
 }  
  
 if (k == -1) {  
 System.*out*.println("\nConnection error\nReturned " + k);  
 break;  
 }  
  
 response += (char) k;  
  
 if (response.endsWith("PSTOP")) {  
 System.*out*.print("\n" + response);  
 break;  
 }  
 }  
  
 //Check message  
 message = response.substring(31, 47).getBytes();  
 receivedFCS = Integer.*parseInt*( response.substring(49, 52) );  
 calculatedFCS = message[0];  
  
 for (int i=1; i<message.length; i++){ //Calculate FCS  
 calculatedFCS = calculatedFCS ^ message[i];  
 }  
  
 if(calculatedFCS == receivedFCS){ // Message is correct  
 delay = System.*currentTimeMillis*() - send;  
 responseFile.write((delay+"\n").getBytes());  
 System.*out*.print(" " + delay + "ms");  
  
 repeatsFile.write( (repeats + "\n").getBytes() );  
 repeats = 0;  
  
 code = ack;  
  
 //Finish test if time has passed.  
 if (System.*currentTimeMillis*() - start >= duration) {  
 addToLog("\nFinished ARQ test ("+  
 LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*(DATETIME\_PATTERN))+  
 ")\nACK code sent " + acksSent + " times" +  
 "\nNACK code sent " + nacksSent + " times");  
 System.*out*.println("\n---STOP---");  
 break;  
 }  
  
 } else{ // Message is incorrect  
 code = nack;  
 }  
  
 }  
 } catch(Exception e){  
 System.*out*.println(e);  
 } finally{ // Close files  
 if(responseFile!=null){  
 try {  
 responseFile.close();  
 } catch (Exception e){  
 System.*out*.println(e);  
 }  
 }  
 if(repeatsFile!=null){  
 try {  
 repeatsFile.close();  
 } catch (Exception e){  
 System.*out*.println(e);  
 }  
 }  
 }  
 }  
  
  
 */\*\*  
 \* Getters  
 \*/* public String getLog(){  
 return log;  
 }  
  
  
 */\*\*  
 \* Setters  
 \*\*/* public void addToLog(char character){  
 log += character;  
 }  
 public void addToLog(String string){  
 log += string;  
 }  
 public void addToLog(int integer){  
 log += integer;  
 }  
 public void setModemSpeed(int speed){  
 modem.setSpeed(speed);  
 }  
  
  
 */\*\*  
 \* Method writeLogFile  
 \* <br><br>  
 \* Write log string to the end of log.txt file  
 \*/* public void writeLogFile() {  
 FileWriter writer = null;  
  
 try {  
 writer = new FileWriter("log.txt", true);  
 writer.write(getLog() + "\n-------------------------END-------------------------\n\n\n\n");  
 } catch (Exception e) {  
 System.*out*.println(e);  
 } finally{  
 if(writer != null){  
 try {  
 writer.close();  
 log = "";  
 } catch (Exception e){  
 System.*out*.println(e);  
 }  
 }  
 }  
 }  
  
 */\*\*  
 \* Method close  
 \* <br><br>  
 \* Close connection  
 \*/* public void close(){  
 while(!modem.close()){  
 System.*out*.println("Can't close modem!");  
 }  
 }  
  
}

Δίκτυα Υπολογιστών Ι  
Ρόντος Έκτορας – Θωμάς  
ΑΕΜ: 9477

## UserApplication.java

import java.util.Scanner;  
  
public class UserApplication {  
  
 public static void main(String[] param) {  
 Scanner scanner = new Scanner(System.*in*);  
 String action, requestCode, secondaryCode = "";  
 int testDuration = 4, timeBetweenTraces = 10;  
 VirtualModem virtualModem;  
  
 System.*out*.println("Welcome!");  
 do { //Ask for action {E, M, G, P}  
 System.*out*.println("Choose action by pressing the corresponding letters:");  
 System.*out*.println("\t<E> Echo request response time test.");  
 System.*out*.println("\t<M> Download and save image without errors.");  
 System.*out*.println("\t<G> Download and save image with errors.");  
 System.*out*.println("\t<P> Download and save image with GPS traces.");  
 System.*out*.println("\t<Q> ARQ Test.");  
 action = scanner.nextLine();  
 } while(!action.equals("E") && !action.equals("M") && !action.equals("G") && !action.equals("P") && !action.equals("Q")); //Repeat if invalid action is given  
  
 System.*out*.println("Write the request code (without special chars).");  
 requestCode = scanner.nextLine();  
  
 switch (action){ //Secondary info needed for some cases.  
 case "E":  
 System.*out*.println("Duration of test? (m)");  
 testDuration = (new Scanner(System.*in*)).nextInt();  
 break;  
 case "P":  
 System.*out*.println("Follow-up code (without special chars).");  
 secondaryCode = (new Scanner(System.*in*)).nextLine();  
 System.*out*.println("Time between traces? (s)");  
 timeBetweenTraces = (new Scanner(System.*in*)).nextInt();  
 break;  
 case "Q":  
 System.*out*.println("NACK code (without special chars).");  
 secondaryCode = (new Scanner(System.*in*)).nextLine();  
 System.*out*.println("Duration of test? (m)");  
 testDuration = (new Scanner(System.*in*)).nextInt();  
 break;  
 }  
  
 System.*out*.println("Set virtual modem speed. (kbps)");  
 virtualModem = new VirtualModem((new Scanner(System.*in*)).nextInt(), 2000);  
  
 virtualModem.read("\r\n\n\n"); //Read welcoming message from Ithaki  
  
 switch (action){ //Start chosen action  
 case "E":  
 virtualModem.echoPacketTest(requestCode+"\r", testDuration \* 1000 \* 60);  
 break;  
 case "M":  
 virtualModem.write(requestCode+"\r");  
 virtualModem.saveImage("E1");  
 break;  
 case "G":  
 virtualModem.write(requestCode+"\r");  
 virtualModem.saveImage("E2");  
 break;  
 case "P":  
 virtualModem.write(requestCode + secondaryCode + "\r");  
 virtualModem.write(requestCode + virtualModem.getGpsTraces(timeBetweenTraces) + "\r");  
 virtualModem.saveImage("M1");  
 break;  
 case "Q":  
 virtualModem.arqTest(requestCode+"\r", secondaryCode+"\r", testDuration \* 1000 \* 60);  
 break;  
 }  
  
 // Save log and close modem  
 virtualModem.writeLogFile();  
 virtualModem.close();  
 }  
  
}

Δίκτυα Υπολογιστών Ι  
Ρόντος Έκτορας – Θωμάς  
ΑΕΜ: 9477

## Project Structure

Το log, αποθηκευόταν στο αρχείο log.txt, στον root φάκελο. Επιπλέον, υπήρχαν οι φάκελοι data, για τα .csv αρχεία με τις τιμές για τα γραφήματα και img για τις εικόνες.