Δίκτυα Υπολογιστών I - Source Code

Στεφανίδης Ιωάννης

AEM: 9587

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;

import ithakimodem.*;

public class userApplication {
    public static final String FILES_PATH = "../graphs/session1";
    public static final String ECHO = "E7901\r";
    public static final String IMAGE_ERROR_FREE = "M2303\r";
    public static final String IMAGE_WITH_ERROR = "G4482\r";
    public static final String GPS = "P9352\r";
    public static final String ACK = "Q7490\r";
    public static final String NACK = "R1469\r";
    private static Modem modem = new Modem();

    public static void main(String[] args) throws Exception {
        modem.setSpeed(1000);
        modem.setTimeout(2000);
        modem.open("ithaki");
        readUntilSequence();

        long startOfFourMinutes = System.currentTimeMillis();
        ArrayList<Packet> packets = new ArrayList<Packet>();
        System.out.println("Getting packets without errors for 4 minutes: \n");
        while (System.currentTimeMillis() < startOfFourMinutes + (4 * 60 * 1000)) {
            packets.add(echoPacket());
        }
        exportDataToFile(packets, "packets_without_errors");

        System.out.println("Getting image without errors: \n");
```

```
    getImage(IMAGE_ERROR_FREE);

    System.out.println("Getting image with errors: \n");
    getImage(IMAGE_WITH_ERROR);

    System.out.println("Getting image with 4 gps points: \n");
    gps();

    System.out.println("Getting packets with errors for 4 minutes: \n");
    startOfFourMinutes = System.currentTimeMillis();
    packets = new ArrayList<Packet>();
    while (System.currentTimeMillis() < startOfFourMinutes + (4 * 60 * 1000)) {
        packets.add(arq());
    }
    exportDataToFile(packets, "packets_with_errors");

    System.out.println("\nmodem closed -> " + modem.close());
    System.exit(0);
}

/**
 *
 * @param numOfChars the number of chars that the packet has
 * @return char[] with packet's chars
 */
private static char[] readPacket(int numOfChars) {
    char[] chars = new char[numOfChars];
    int k;
    for (int i = 0; i < numOfChars - 1; i++) {
        try {
            k = modem.read();

            if (k == -1) {
                System.out.println("modem sent -1");
                break;
            }

            chars[i] = (char) k;

            // System.out.print((char) k);
        } catch (Exception x) {
            System.out.print(x.getMessage());
            break;
        }
    }
    return chars;
```

```
}

private static void runCommand(String command) {
    byte[] commandInBytes = null;
    try {
        commandInBytes = command.getBytes("US-ASCII");
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    boolean commandSent = modem.write(commandInBytes);
    if (!commandSent)
        System.out.println("Error sending the command " + command);
}

/**
 * read until \r\n\n\n
 *
 * @param modem the modem to read from
 */
private static void readUntilSequence() {
    int k;
    for (int i = 0; i != 4;) {
        try {
            k = modem.read();
            if (k == -1) {
                System.out.println("modem sent -1");
                break;
            }
            char a = (char) k;
            System.out.print(a);
            // check for \r\n\n\n sequence
            if (i >= 1) {
                if (a == '\n')
                    i++;
                else
                    i = 0;
            }
            if (a == '\r') {
                i = 1;
            }

        } catch (Exception x) {
            System.out.print(x.getMessage());
            break;
        }
    }
```

```
}

/**
 * Prints packets from ithaki server and returns it as Packet object
 *
 * @throws IOException
 */
private static Packet echoPacket() throws IOException {
    modem.setSpeed(80000);
    long startTime = System.currentTimeMillis();
    runCommand(ECHO);
    char[] chars = readPacket(36);
    long elapsedTime = System.currentTimeMillis() - startTime;
    Packet packet = new Packet(new String(chars), elapsedTime);
    System.out.println(packet.toString());
    return packet;
}

/**
 *
 * @param packets  Packets to export
 * @param filename filename to save to
 * @throws IOException
 */
private static void exportDataToFile(ArrayList<Packet> packets, String
filename) throws IOException {
    File dataFile = new File(FILES_PATH + filename + ".csv");
    FileWriter writer = new FileWriter(dataFile);
    for (int i = 0; i < packets.size(); i++) {
        if (packets.get(i).noIdea != null)
            writer.write(packets.get(i).packetString + ";" +
            packets.get(i).repeate + ";"
                    + packets.get(i).responseTime + "\n");
        else
            writer.write(packets.get(i).packetString + ";" +
            packets.get(i).responseTime + "\n");
    }
    writer.close();
}

/**
 *
 * @param requestCode the image request code
 * @throws InterruptedException
 */
private static void getImage(String requestCode) throws InterruptedException {
```

```
modem.setSpeed(80000);
ArrayList<Byte> imageByte = new ArrayList<Byte>();
try {
    runCommand(requestCode);
    int k;

    boolean ffRead = false;
    System.out.println("loading started");
    for (;;) {
        try {
            k = modem.read();
            if (k == -1) {
                System.out.println("modem sent -1");
                break;
            }

            imageByte.add((byte) k);

            if (ffRead) {
                String last = Integer.toHexString(k);
                if (last.equals("d9")) {
                    System.out.println("\nfound 0xff 0xd9");
                    break;
                }
            }

            if (Integer.toHexString(k).equals("ff"))
                ffRead = true;
            else
                ffRead = false;

        } catch (Exception x) {
            System.out.print("stopped by Exception");
            break;
        }
    }
    System.out.println("loading stopped");
    byte[] img = new byte[imageByte.size()];
    for (int j = 0; j < imageByte.size(); j++)
        img[j] = imageByte.get(j);

    String imageString = FILES_PATH + "image_" +
    requestCode.replace('\r', '.') + "jpg";
    File outputfile = new File(imageString);
    FileOutputStream os = new FileOutputStream(outputfile);
    os.write(img);
```

```
        os.close();
        System.out.println("Image saved at " + imageString);

    } catch (IOException e) {
        System.out.println("An error occurred.");
        e.printStackTrace();
    }

}

/**
 * @param output is where to store all the output from ithaki
 */
private static void gpsRead(StringBuilder output) {
    int k;
    for (;;) {
        try {
            k = modem.read();
            output.append((char) k);
            if (k == -1) {
                System.out.println("modem sent -1");
                break;
            }

            System.out.print((char) k);

            if (output.length() >= 26) {
                if (output.substring(output.length() - 26).equals("STOP
                ITHAKI GPS TRACKING\r\n")) {
                    break;
                }
            }
        } catch (Exception x) {
            System.out.print(x.getMessage());
            break;
        }
    }
}

/**
 * create image with 4 gps points
 */
private static void gps() {
    //creating a stringbuild to store all 4 gps outputs and then create
    GPGGA objects
    StringBuilder output = new StringBuilder();
```

```
    modem.setSpeed(1000);

    //Manually select PPPP to get better image

    // Session 1
    runCommand(GPS.replaceAll("\r", "R=1000001\r"));
    gpsRead(output);
    runCommand(GPS.replaceAll("\r", "R=1010001\r"));
    gpsRead(output);
    runCommand(GPS.replaceAll("\r", "R=1011001\r"));
    gpsRead(output);
    runCommand(GPS.replaceAll("\r", "R=1012001\r"));
    gpsRead(output);

    // Session 2 = Session 1 + 10
    //runCommand(GPS.replaceAll("\r", "R=1001101\r"));
    //gpsRead(output);
    //runCommand(GPS.replaceAll("\r", "R=1011001\r"));
    //gpsRead(output);
    //runCommand(GPS.replaceAll("\r", "R=1012001\r"));
    //gpsRead(output);
    //runCommand(GPS.replaceAll("\r", "R=1013001\r"));
    //gpsRead(output);

    ArrayList<GPGGA> gpggaList = new ArrayList<GPGGA>();

    for (int i = 0; i < output.length(); i++) {
        if (output.charAt(i) == '$') {
            int start = i;
            while (output.charAt(i) != '\r') {
                i++;
            }
            int stop = i;
            GPGGA aGpgga = new GPGGA(output.substring(start, stop));
            gpggaList.add(aGpgga);
        }
    }

    getGpsImage(gpggaList);
}

/**
 * @param gpggaList GPGGA objects to create the T parameters
 */
private static void getGpsImage(ArrayList<GPGGA> gpggaList) {
    StringBuilder tPar = new StringBuilder();
```

```
        for (int i = 0; i < gpggaList.size(); i++) {
            tPar.append(gpggaList.get(i).getT());
        }
        tPar.append("\r");
        String command = GPS.replaceAll("\r", tPar.toString());
        System.out.println(command);
        try {
            getImage(command);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    /**
     * ARQ system to get packet without error
     *
     * @return the correct packet
     */
    private static Packet arq() {
        modem.setSpeed(80000);
        int packetRepeat = 0;
        long startTime = System.currentTimeMillis();
        runCommand(ACK);
        char[] chars = readPacket(59);
        long elapsedTime = System.currentTimeMillis() - startTime;
        Packet aPacket = new Packet(new String(chars), elapsedTime, packetRepeat);

        while (!aPacket.isFCScorrect()) {
            System.out.println("\nwrong FCS");
            packetRepeat++;
            startTime = System.currentTimeMillis();
            runCommand(NACK);
            chars = readPacket(59);
            elapsedTime = elapsedTime + System.currentTimeMillis() - startTime;
            aPacket = new Packet(new String(chars), elapsedTime, packetRepeat);
        }
        System.out.println("\ncorrect FCS");
        System.out.println(aPacket.toString());
        return aPacket;
    }
}

class Packet {
    final String packetString;
    final long responseTime;
    String noIdea;
```

```
int fcs;
int repeate;

/**
 * Packet with error
 *
 * @param packetString
 * @param responseTime
 * @param repeate
 */
Packet(String packetString, long responseTime, int repeate) {
    this.packetString = packetString;
    String[] splitted = packetString.split(" ");
    this.responseTime = responseTime;

    this.repeate = repeate;
    this.noIdea = splitted[4];
    this.fcs = Integer.parseInt(splitted[5]);
}

/**
 * Packet without error
 *
 * @param packetString
 * @param responseTime
 */
Packet(String packetString, long responseTime) {
    this.packetString = packetString;
    this.responseTime = responseTime;

}

@Override
public String toString() {
    if (noIdea != null) {
        return packetString + " with response time = " + responseTime + "ms"
        + " number of trys: " + repeate;
    } else
        return packetString + " with response time = " + responseTime + "ms";
}

public boolean isFCScorrect() {
    int x = noIdea.charAt(1);
    for (int i = 2; i < 17; i++) {
        x = x ^ noIdea.charAt(i);
    }
```

```
        return x == fcs;
    }
}

class GPGGA {
    private String ithakiOutput;

    private int latDegrees;
    private int latMinutes;
    private int latSeconds;

    private int longDegrees;
    private int longMinutes;
    private int longSeconds;

    final String time;

    /**
     * @param ithakiOutput The exact output from ithaki server
     */
    GPGGA(String ithakiOutput) {
        this.ithakiOutput = ithakiOutput;
        String[] info = ithakiOutput.split(",");

        this.time = info[1];

        String lat = info[2];
        String longt = info[4];

        this.latDegrees = Integer.parseInt(lat.substring(0, 2));
        this.latMinutes = Integer.parseInt(lat.substring(2, 4));
        this.latSeconds = (int) (60 * Double.parseDouble(lat.substring(5, 9)) / 10000);

        this.longDegrees = Integer.parseInt(longt.substring(0, 3));
        this.longMinutes = Integer.parseInt(longt.substring(3, 5));
        this.longSeconds = (int) (60 * Double.parseDouble(longt.substring(6, 10)) / 10000);
    }

    /**
     * @return T parameter to get image from ithaki
     */
    public String getT() {
        String t = "T=" + longDegrees + longMinutes + longSeconds + latDegrees +
        latMinutes + latSeconds;
        return t;
    }
```

```
    @Override
    public String toString() {
        return ithakiOutput;
    }
}
```