package com.company;  
import ithakimodem.Modem;  
import java.io.IOException;  
import java.io.OutputStream;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.lang.System;  
import java.io.FileOutputStream;  
import java.time.Duration;  
import java.time.Instant;  
import org.apache.poi.ss.usermodel.Cell;  
import org.apache.poi.ss.usermodel.Row;  
import org.apache.poi.xssf.usermodel.XSSFCell;  
import org.apache.poi.xssf.usermodel.XSSFRow;  
import org.apache.poi.xssf.usermodel.XSSFSheet;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
  
  
  
public class virtualModem {  
 public static void main(String[] param) throws IOException {  
 (new virtualModem()).demo();  
 }  
 public void demo() throws IOException {  
 int k;  
 Modem modem;  
 modem = new Modem();  
 modem.setSpeed(80000);  
 modem.setTimeout(2000);  
 modem.open("ithaki");  
 for (; ; ) {  
 k = modem.read();  
 if (k == -1) break;  
 System.*out*.print((char) k);  
 }  
  
 //Print packages for 4.2 minutes and return the G1 graph of response time of each one of them  
  
  
 Instant firstPacket = Instant.*now*();  
 String echoRequestCode = "E4224\r";  
 int byteCounter1 = 0;  
 int packageCounter1 = 0;  
 long[] responseTimeArray1 = new long[6000];  
 long initialTime1 = 0;  
 long finalTime1 = 0;  
 long average1 = 0;  
 while (Duration.*between*(firstPacket,Instant.*now*()).toSeconds() < 252){  
 String message = "";  
 modem.write(echoRequestCode.getBytes());  
 initialTime1 = System.*currentTimeMillis*();  
 for (; ; ) {  
 k = modem.read();  
 byteCounter1++;  
 if (k == -1) break;  
 message += (char) k;  
 System.*out*.print((char) k);  
 if(message.contains("PSTOP")){  
 finalTime1 = System.*currentTimeMillis*();  
 System.*out*.print("\nResponse Time = " + (finalTime1-initialTime1) + "ms");  
 average1 += finalTime1-initialTime1;  
 responseTimeArray1[packageCounter1] = finalTime1 - initialTime1;  
 break;  
 }  
 }  
 System.*out*.println("\n");  
 packageCounter1 ++;  
 }  
 System.*out*.print("Number of packages = " + packageCounter1 + "\n");  
 System.*out*.print("Average Response time = " + (average1/packageCounter1) + "\n");  
  
 //Create an excel file with your response time results  
  
 XSSFWorkbook workbook = new XSSFWorkbook();  
 XSSFSheet sheet = workbook.createSheet("Diagrams");  
 XSSFRow row = sheet.createRow(0);  
 for(int a = 0; a < packageCounter1; a++){  
 XSSFCell cell = row.createCell(a);  
 cell.setCellValue(responseTimeArray1[a]);  
 }  
 FileOutputStream outputStream = new FileOutputStream("DiagramsG1.xlsx");  
 workbook.write(outputStream);  
 workbook.close();  
  
  
 //Print an error-less image  
  
 String imageRequestCode = "M4036\r";  
 modem.write(imageRequestCode.getBytes());  
 int[] imageArray1 = new int[80000];  
 byte[] Array1 = new byte[80000];  
 int i = 0;  
 for(;;){  
 int a = modem.read();  
 imageArray1[i] = a;  
 Array1[i] = (byte) a;  
 i ++;  
 if((a == 217) && (imageArray1[i-2] == 255) && (i >= 2)) break;  
 }  
 Path image1 = Path.*of*("C:\\Users\\konst\\errorlessImage.jpg");  
 Files.*write*(image1,Array1);  
  
 //Print a gps Image with 5 pins  
  
 String gpsRequestCode = "P5070R=1003030\r";  
 modem.write(gpsRequestCode.getBytes());  
 for(;;){  
 String gpsmessage = "";  
 k = modem.read();  
 if (k == -1) break;  
 gpsmessage += (char) k;  
 System.*out*.print((char) k);  
 if (gpsmessage.contains("STOP ITHAKI GPS TRACKING")) {  
 break;  
 }  
 }  
 String gpsImageCode = "P5070T=225735403737T=225735403737T=225734403738T=225733403738T=225732403739\r";  
 modem.write(gpsImageCode.getBytes());  
 int[] imageArray2 = new int[130000];  
 byte[] Array2 = new byte[130000];  
 int t = 0;  
 for(;;){  
 int a = modem.read();  
 imageArray2[t] = a;  
 Array2[t] = (byte) a;  
 t ++;  
 if((a == 217) && (imageArray2[t-2] == 255) && (t>= 2)) break;  
 }  
 Path imageGps = Path.*of*("C:\\Users\\konst\\gpsImage.jpg");  
 Files.*write*(imageGps,Array2);  
  
 //Print an image with errors  
  
 String errorImageRequestCode = "G5922\r";  
 modem.write(errorImageRequestCode.getBytes());  
 int[] errorImageArray = new int[80000];  
 byte[] errorArray = new byte[80000];  
 int n = 0;  
 for(;;){  
 int a = modem.read();  
 errorImageArray[n] = a;  
 errorArray[n] = (byte) a;  
 n ++;  
 if((a == 217) && (errorImageArray[n-2] == 255) && (n >= 2)) break;  
 }  
 Path errorImage = Path.*of*("C:\\Users\\konst\\errorImage.jpg");  
 Files.*write*(errorImage,errorArray);  
  
 //Ack and Nack codes  
  
 int probability = 0;  
 int packageCounter2 = 1;  
 long finalTime2 = 0;  
 long initialTime2 = 0;  
 long average2 = 0;  
 long[] responseTimeArray2 = new long[6000];  
 String ackCode = "Q6997\r";  
 String nackCode = "R0214\r";  
 int ackTotalCounter = 1;  
 int nackTotalCounter = 0;  
 int nackCounter = 0;  
 int rightPackageCounter = 0;  
 int[] nackNumberOfEveryPackage = new int[6000];  
 modem.write(ackCode.getBytes());  
 Instant loopStart = Instant.*now*();  
 while(Duration.*between*(loopStart,Instant.*now*()).toSeconds() < 252){  
 int l = 0;  
 int[] errorCheck = new int[16];  
 int[] fcs = new int[3];  
 initialTime2 = System.*currentTimeMillis*();  
 for (; ; ) {  
 k = modem.read();  
 l ++;  
 if (k == -1) break;  
 System.*out*.print((char) k);  
 if(l >= 32 && l <= 47){  
 errorCheck[l-32] = k;  
 }  
 if(l >= 50 && l <= 52){  
 fcs[l-50] = k;  
 }  
 if(l == 58){  
 finalTime2 = System.*currentTimeMillis*();  
 System.*out*.print("\nResponse Time = " + (finalTime2-initialTime2) + "ms");  
 average2 += finalTime2-initialTime2;  
 responseTimeArray2[packageCounter2] = finalTime2-initialTime2;  
 break;  
 }  
 }  
 System.*out*.println();  
 int xor = 0;  
 xor = computeXOR(errorCheck);  
 int fcsnum = (fcs[0]-48)\*100 + (fcs[1]-48)\*10 + (fcs[2]-48);  
 if(xor == fcsnum){  
 modem.write(ackCode.getBytes());  
 probability ++;  
 ackTotalCounter ++;  
 nackNumberOfEveryPackage[rightPackageCounter] = nackCounter;  
 rightPackageCounter++;  
 nackCounter = 0;  
 }  
 else{  
 modem.write(nackCode.getBytes());  
 nackTotalCounter++;  
 nackCounter++;  
 }  
 System.*out*.println("Package number = " + packageCounter2);  
 packageCounter2 ++;  
  
 }  
 System.*out*.println("Number of packs = " + packageCounter2);  
 System.*out*.print("Average Response time = " + (average2/packageCounter2) + "\n");  
 System.*out*.println("Acks = " + ackTotalCounter + " and Nacks = " + nackTotalCounter);  
 System.*out*.println("Prob = " + probability);  
 System.*out*.println("Percentage of correct packs = " + ((float)probability/(float)packageCounter2)\* 100.0 + "%");  
 System.*out*.println();  
  
 //Create an excel file with your response time results  
  
 XSSFWorkbook workbook2 = new XSSFWorkbook();  
 XSSFSheet sheet2 = workbook2.createSheet("Diagrams2");  
 XSSFRow row2 = sheet2.createRow(0);  
 for(int a = 0; a < packageCounter2; a++){  
 XSSFCell cell2 = row2.createCell(a);  
 cell2.setCellValue(responseTimeArray2[a]);  
 }  
 FileOutputStream outputStream2 = new FileOutputStream("DiagramsG2.xlsx");  
 workbook2.write(outputStream2);  
 workbook2.close();  
  
  
 XSSFWorkbook workbook3 = new XSSFWorkbook();  
 XSSFSheet sheet3 = workbook3.createSheet("Diagrams3");  
 XSSFRow row3 = sheet3.createRow(0);  
 XSSFRow row4 = sheet3.createRow(1);  
 for(int a = 0; a < rightPackageCounter; a++){  
 XSSFCell cell3 = row3.createCell(a);  
 XSSFCell cell4 = row4.createCell(a);  
 //cell4.setCellValue();  
 cell3.setCellValue(nackNumberOfEveryPackage[a]);  
 }  
 FileOutputStream outputStream3 = new FileOutputStream("DiagramsG3.xlsx");  
 workbook3.write(outputStream3);  
 workbook3.close();  
 double ber = computeBER(ackTotalCounter,nackTotalCounter);  
 System.*out*.println("ber = " + ber);  
 modem.close();  
 }  
  
 //Compute the xor of consecutive elements of an array  
  
 public int computeXOR(int[] errorCheck){  
 int xor = 0;  
 for(int i = 0; i < 16; i++){  
 xor = (xor)^(errorCheck[i]);  
 }  
 return xor;  
 }  
 //Calculate the BER  
  
 public double computeBER(int ackTotalCounter,int nackTotalCounter){  
 double ber = 0;  
 double p = ((double)(ackTotalCounter))/((double)(ackTotalCounter+nackTotalCounter));  
 double n = ((double) 1) / ((double) 128);  
 ber = 1.0 - Math.*pow*(p,n);  
 return ber;  
 }  
}