**Name:** Hung Viet Luu

**Link to Project:** <https://github.com/hvluu/CS380/tree/master/Projects/Project2>

***PhysLayerClient.java***

*import java.io.IOException;*

*import java.net.Socket;*

*import java.net.UnknownHostException;*

*public class PhysLayerClient*

*{*

*private static Socket socket;*

*public static void main(String[] args)*

*{*

*connect();*

*}*

*/\*\**

*\* Connects the client to the server and*

*\* creates a Listener thread.*

*\*/*

*public static void connect()*

*{*

*String hostName = "18.221.102.182";*

*int portNumber = 38002;*

*try*

*{*

*socket = new Socket(hostName, portNumber);*

*new Connection(socket).start();*

*System.out.println("Connected to server.");*

*}*

*catch (UnknownHostException e)*

*{*

*System.err.println("ERROR: Unknown host " + hostName + ".");*

*}*

*catch (IOException e)*

*{*

*System.err.println("ERROR: Could not connect to " + hostName + ".");*

*}*

*}*

*/\*\**

*\* Disconnects the client from the server.*

*\*/*

*public static void disconnect()*

*{*

*try*

*{*

*socket.close();*

*System.out.println("Disconnected from server.");*

*}*

*catch (IOException e)*

*{*

*System.err.println("ERROR: " + e.getMessage());*

*}*

*}*

*}*

**Connection.java**

*import java.io.\*;*

*import java.net.Socket;*

*import java.util.HashMap;*

*public class Connection extends Thread*

*{*

*public volatile static boolean endThread = false;*

*private Socket socket = null;*

*public Connection(Socket socket)*

*{*

*super("Connecting Thread");*

*this.socket = socket;*

*}*

*/\*\**

*\* Override run() function in the Thread class.*

*\* The function handles the communication between the server and the client.*

*\*/*

*@Override*

*public void run()*

*{*

*final int PREAMBLE\_SIZE = 64;*

*final int DATA\_SIZE = 32;*

*byte[] data;*

*float baseline = calculateBaseline(PREAMBLE\_SIZE);*

*System.out.println("Baseline established from preamble: " + baseline);*

*System.out.print("Received " + DATA\_SIZE + " bytes: ");*

*data = receiveData(DATA\_SIZE, baseline);*

*System.out.println();*

*respond(data);*

*PhysLayerClient.disconnect();*

*}*

*/\*\**

*\* Reads the preamble and calculates the baseline based*

*\* on an average of the received high and low signals.*

*\*/*

*private float calculateBaseline(final int PREAMBLE\_SIZE)*

*{*

*float baseline = 0.0f;*

*try*

*{*

*for(int i = 0; i < PREAMBLE\_SIZE; i++)*

*baseline += socket.getInputStream().read();*

*baseline /= PREAMBLE\_SIZE;*

*}*

*catch (IOException e)*

*{*

*System.err.println("ERROR: " + e.getMessage());*

*}*

*return baseline;*

*}*

*/\*\**

*\* HashMap of the 4B/5B conversion table.*

*\*/*

*private HashMap<String, String> fourBfiveBTable()*

*{*

*HashMap<String, String> hashMap = new HashMap<>();*

*hashMap.put("11110", "0000");*

*hashMap.put("01001", "0001");*

*hashMap.put("10100", "0010");*

*hashMap.put("10101", "0011");*

*hashMap.put("01010", "0100");*

*hashMap.put("01011", "0101");*

*hashMap.put("01110", "0110");*

*hashMap.put("01111", "0111");*

*hashMap.put("10010", "1000");*

*hashMap.put("10011", "1001");*

*hashMap.put("10110", "1010");*

*hashMap.put("10111", "1011");*

*hashMap.put("11010", "1100");*

*hashMap.put("11011", "1101");*

*hashMap.put("11100", "1110");*

*hashMap.put("11101", "1111");*

*return hashMap;*

*}*

*/\*\**

*\* Receives and decodes the data.*

*\*/*

*private byte[] receiveData(final int DATA\_SIZE, float baseline)*

*{*

*// The size is multiplied by 2 because*

*// we receives the bytes in halves.*

*String[] receivedData = new String[DATA\_SIZE \* 2];*

*byte[] data = new byte[DATA\_SIZE];*

*try*

*{*

*// Boolean variables to compare the 2 signals later*

*// pSignal = previous signal*

*// cSignal = current signal*

*boolean pSignal = false;*

*boolean cSignal;*

*String fiveBits;*

*HashMap<String, String> fourBfiveBTable = fourBfiveBTable();*

*for(int i = 0; i < receivedData.length; i++)*

*{*

*fiveBits = "";*

*for(int j = 0; j < 5; j++)*

*{*

*cSignal = socket.getInputStream().read() > baseline;*

*// Decoding NRZI signal.*

*// It compares the cSignal with baseline and determine the*

*// correct value.*

*// Then the pSignal and cSignal get compared and*

*// the data bit's value is then determined.*

*if (pSignal == cSignal)*

*fiveBits += "0";*

*else*

*fiveBits += "1";*

*pSignal = cSignal;*

*}*

*receivedData[i] = fourBfiveBTable.get(fiveBits);*

*}*

*String firstHalf;*

*String secondHalf;*

*String completeByteString;*

*int completeByte;*

*// Data reconstruction occurs here.*

*for(int i = 0, j = 0; i < data.length; i++, j += 2)*

*{*

*firstHalf = receivedData[j];*

*secondHalf = receivedData[j + 1];*

*completeByteString = firstHalf + secondHalf;*

*completeByte = Integer.parseInt(completeByteString, 2);*

*System.out.print(Integer.toHexString(completeByte).toUpperCase());*

*data[i] = (byte) completeByte;*

*}*

*}*

*catch(IOException | NumberFormatException e)*

*{*

*System.err.println("ERROR: " + e.getMessage());*

*}*

*return data;*

*}*

*/\*\**

*\* Sends the response to the server.*

*\*/*

*private void respond(byte[] response)*

*{*

*try*

*{*

*socket.getOutputStream().write(response);*

*int serverResponse;*

*if((serverResponse = socket.getInputStream().read()) == 1)*

*System.out.println("Response good.");*

*else*

*System.out.println("Bad response. Server returned " + serverResponse);*

*}*

*catch (IOException e)*

*{*

*System.err.println("ERROR: " + e.getMessage());*

*}*

*}*

*}*