**Name:** Mathilde Moloux

**Mail:** mathilde.moloux2@mail.dcu.ie

**Student Number:** 18102832

**Program:** Study abroad

EE402

Assignment 2

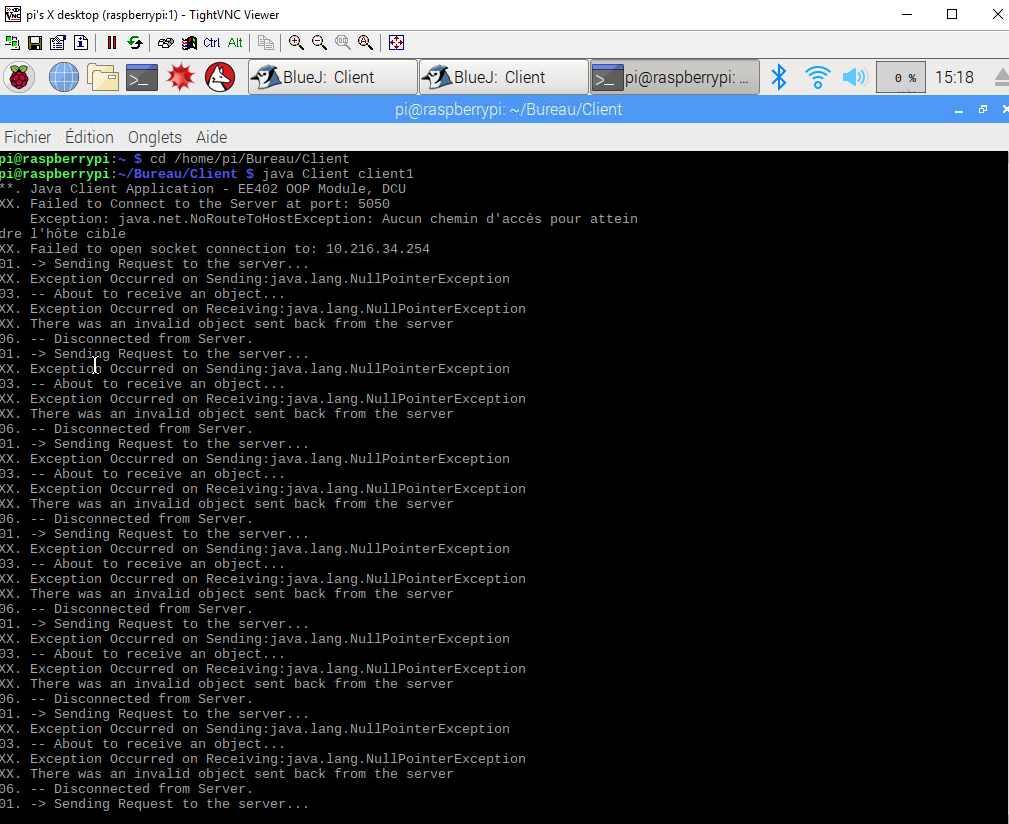
**Java Client/Server Application**

**Report Structure**

December 2018

1. The raspberry pi

First of all, I tried to use a raspberry. I had the code of the Client and a class called TransferData (which read the temperature in a file) on it. However, when I execute the code, I had the following errors:



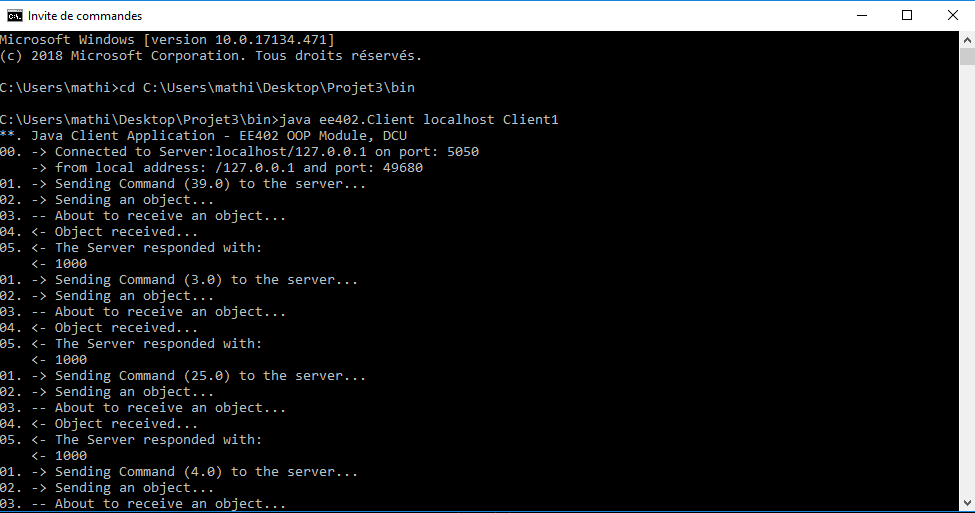
The client couldn’t recognize my IP address despite numerous attempts and I didn’t know why. When I executed my code on a different PC it worked and I had a connection between the client and the server.

As I couldn’t have a connection Client/Server using my own computer with a raspberry, I decided to run locally the client on my computer.

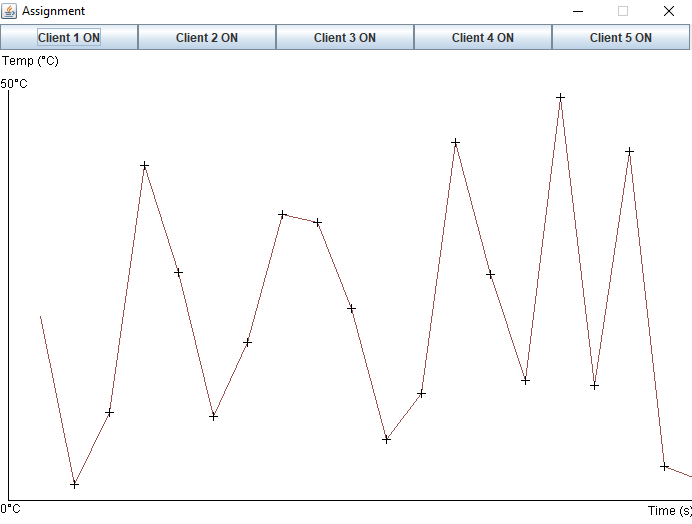
1. The different features in local host

I can do a connection between the client and the server in localhost. The client sends objects to the server (random temperatures) and the server responds by sending a frequency to the client.

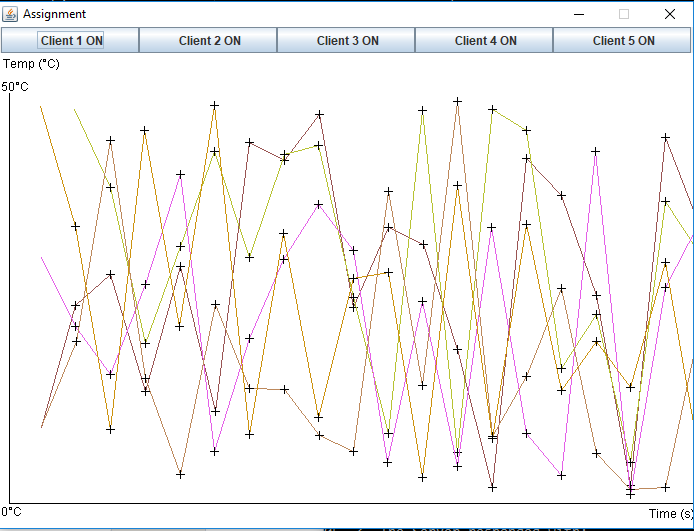
The client is working by taping the command ee402.Client localhost Client1. Client1 is just an example of a name of the client.



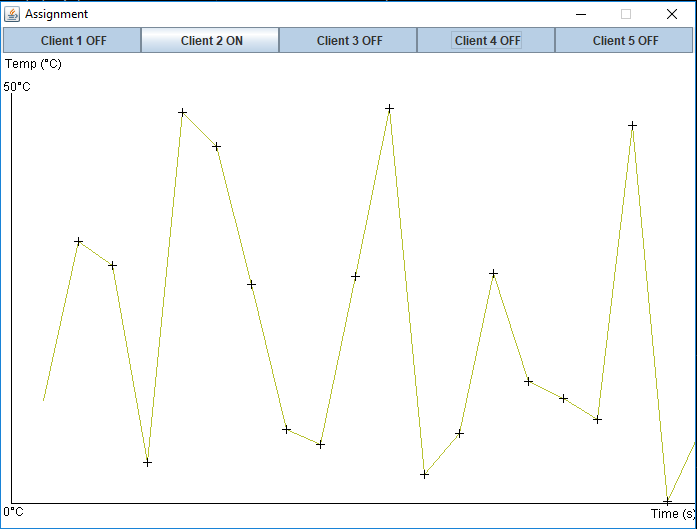
Here the graph of the different temperatures sending by the client, displaying the last 20 readings:



I can also launch several clients. Here I launched five different clients:



I have also toggle buttons. If I click on the button it stops displaying the curve of the client and if I click again it displays the curve.

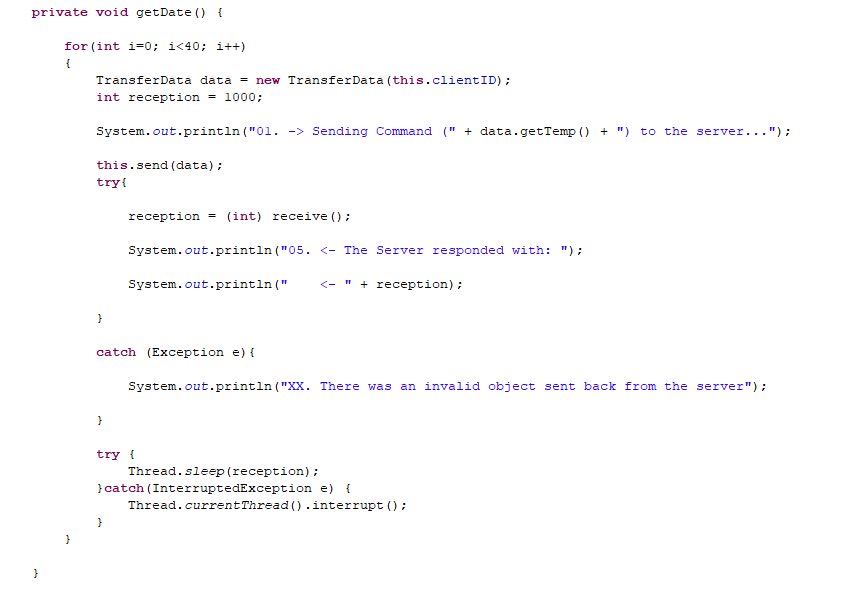


1. The implementation

In my project, there are nine classes.

* The class Client: The client sends a temperature to the server.

In my code, the client sends 40 objects of type TransferData to the server and receive a frequency from the client. Here is the section of the code:



* The class ThreadedServer: It is the main server application. It listens for connection and when there is a connection to the server it creates a ThreadedConnectionHandler object.

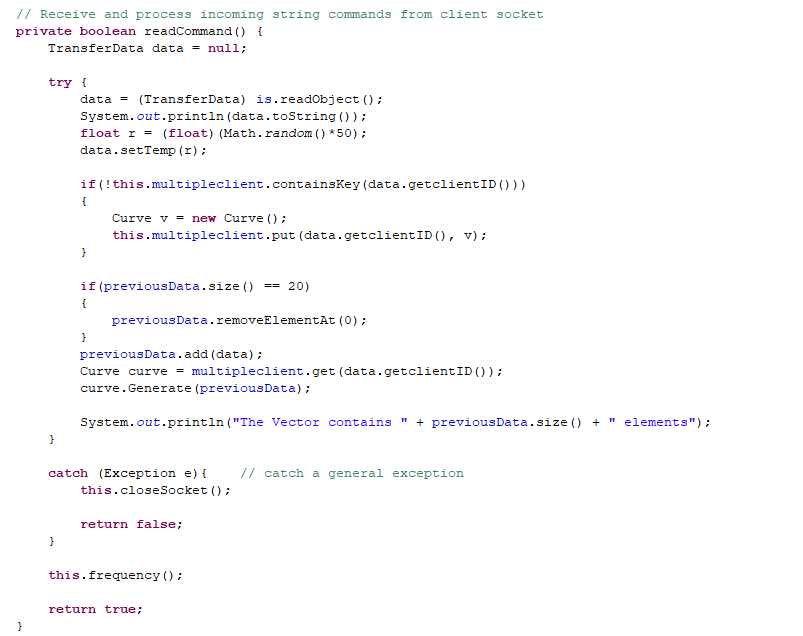
The server sends a frequency to the client.

To have multiple clients, I used a map <String, Curve>. The map stores the name of the client (String) and associate a curve to it.

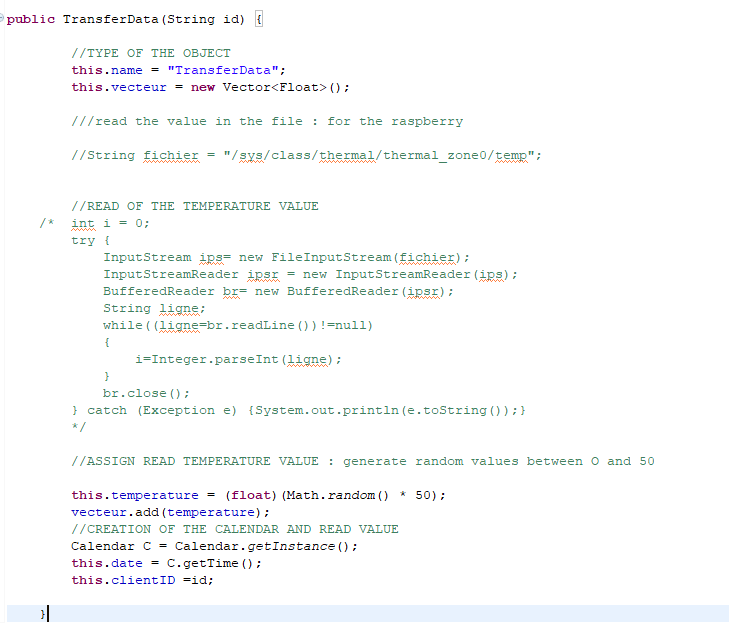
* The class ThreadedConnectionHandler: It handles the requests that arrive to the server.

There are five attributes: One client socket object, one input stream, one output stream. I used a vector to stores the object of type TransfertData and a map that stores the name of the client (String) and associate a curve to it.

The function readCommand() is a boolean. It returns false if it catches a general exception and return true if not. This function receives and process incoming strings commands from client socket. I used a setter to modify the temperature with a random value. When I add a point to my vector, I checked the size of it, remove the oldest element and add a new point. I can’t have more than 20 elements in it. Then, I generate the curve with the data.



* The class TransferData: In this class there is the section of code where it should read the values of the temperature of the raspberry, so I put this section is comments. Instead of, I generate random values between 0 and 50 degrees. Then, I added the values into a vector of float.



* The class Bool: As I can have five clients, I coded a class with five booleans which are initialized to true. I use it in the class GuiApp for the toggle buttons.

I also used boolean in the class Graph in the method paint() which display the curve for the condition:



Client.c1==true verifies that a client exists, p.size()>1 verifies that there is one element in the vector and j==1 is number of the curve.

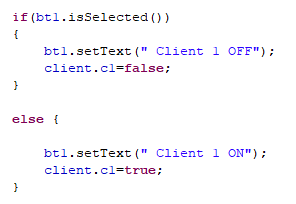
* The class GuiApp: This class is my interface. It is where there is the main of my application.

I created here two panels. One for the button, and one is a combination of the buttons and the curve. The panel for the curve is implemented in Graph class.

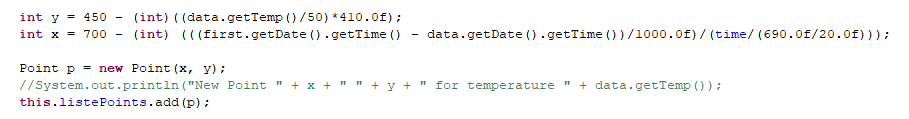
I also created the five toggle buttons that display the curve of a client or not.

If I click on the button the state of the client is changing. I used Boolean.

Here is the code for one button:



* The class Curve: In this class there are three attributes a vector of points, a float for the time and an object of type color. I have a method called addingPoint() that instantiate a point with the coordinate x and y witch are calculated with the following formulae:



Then, I add the points into the vector of points.

* The class Point: This class is for the coordinate x and y of the point of the curve.

To conclude, I don’t have all the features in the project but all I did is functional.

As I never studied java before, this project allowed me to discover a new programming language.

1. Resource page

<https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java>

<https://www.developpez.net/forums/d767224/java/interfaces-graphiques-java/graphisme/2d/dessiner-courbe-point-point/>

<http://ee402.eeng.dcu.ie>