**DCU School of Electronic Engineering   
Assignment Submission**

|  |  |
| --- | --- |
| **Student Name:** | **Eoin Flanagan** |
| **Student Number:** | **59055860** |
| **Programme:** | **Masters Engineering Qualifier  Electronic & Computer Engineering** |
| **Project Title:** | **Assignment 2** |
| **Module code:** | **EE402** |
| **Lecturer:** | **Dr. Derek Molloy** |
| **Project Due Date:** | **14/12/2018** |

Top of Form

Bottom of Form

Introduction 4

Implementation 5

General Overview 5

Each class in depth 6

Q5 6

ThreadedConnectionHandler 6

GraphGUI 6

EmptyClass (PC) 7

EmptyClass (PI) 7

Client 7

Running Program 8

How to run. 10

Conclusion 11

Source Code 13

Q5 Class 13

GraphGUI 29

IntTextField 50

EE402Incrament 51

EmptyClass – PC 52

ThreadedServer 54

ThreadedConnectionHandler 57

Client 63

EmptyClass (Pi) 69

Figure 1 4

Figure 2 5

Figure 3 5

Figure 4 8

Figure 5 9

Figure 6 9

Figure 7 10

Figure 8 11

Figure 9 11

# Introduction

This report serves as a description of the submitted assignment. It will outline how the tasks set out where achieved. It will include screenshots and the source code will be provided at the end.

The project consists of eight classes and one interface. Five of the classes and the interface is used on the laptop/PC. The other two are used on the Raspberry Pi

1. Q5 – main run
2. GraphGUI
3. ThreadedServer
4. ThreadedConnectionHandler
5. EmptyClass (Laptop)
6. EE402Increment (Interface)
7. IntTexField
8. EmptyClass (Raspberry Pi)
9. Server – main run

A general layout of the design can be seen in Figure 1.

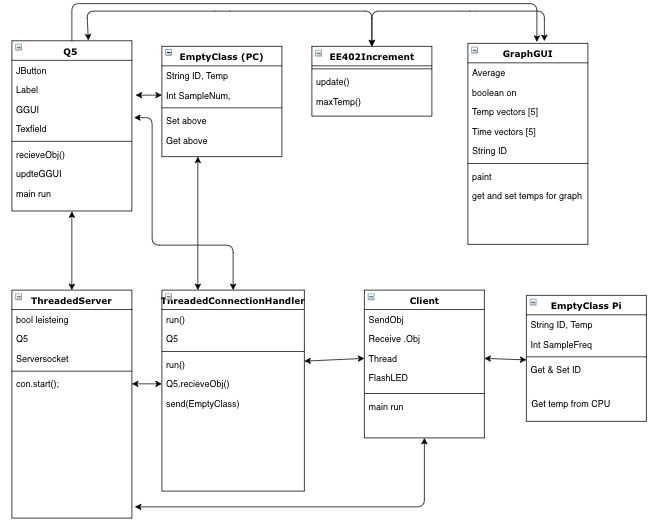


Figure 1

# Implementation

## General Overview

Execution flow/Comms:  
Q5 has the main run function. It creates a Q5 class in its main. Inside the constructor for the Q5 class the GUI and relevant buttons are added. Q5 calls the GraphGUI class which creates the canvas. The buttons and canvas are made visible. Then Q5 calls the ThreadedServer which in creates our ThreadedConnectionHandler con, con is a thread and begins its run state, which then loops a while listening, which continues listening for connection indefinitely. The status of the program at this point can be seen in Figure 2

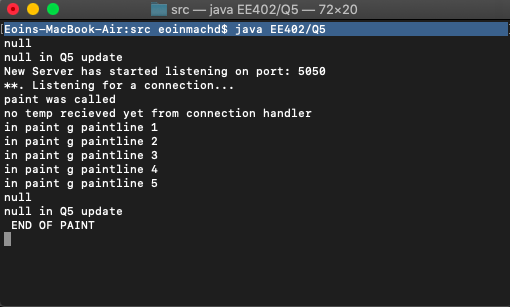
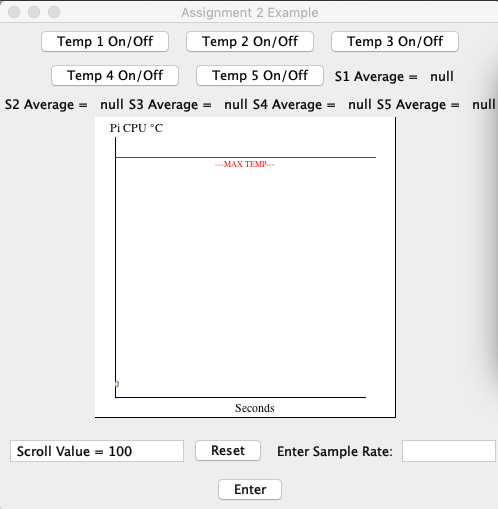


Figure 2

The client contains the main run on the raspberry pi. Its condition so that the user must input an IP address and an ID of the current run client. If credentials are valid the client calls a runApp which loops a running Boolean which uses try and catch to attempt communication with Q5/ThreadedServer/TCH over the socket. A new object of EmptyClass is created and sent every time a iteration of the loop takes place. Iterations are slowed by thread sleeps to control the sample rate. See the Figure 3 for system outputs after initialising the programs. (not using Pi in this case)

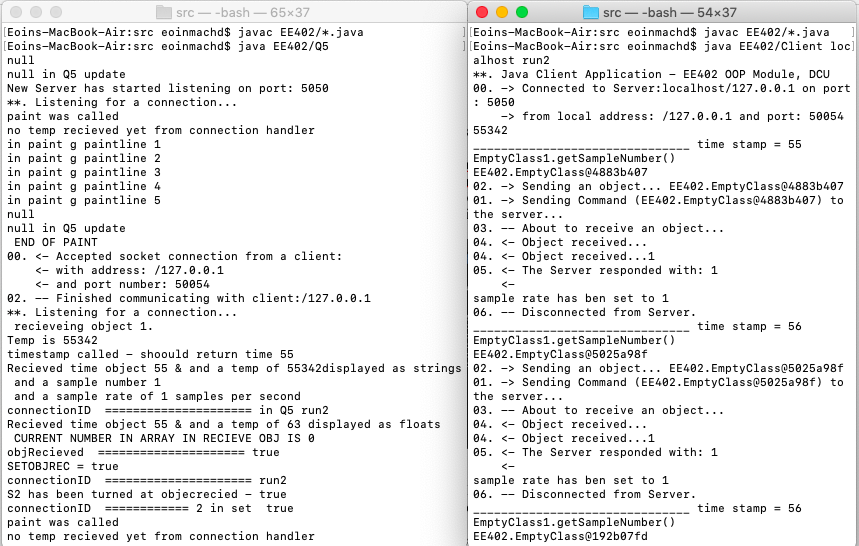


Figure 3

## Each class in depth

### Q5

The project plots the 5 temps on the same graph. Q5 creates the ThreadedServer in its constructor and passes itself. The ThreadedServer creates a ThreadedConnectionHandler ‘con’ and passes a handle to the Q5 in its constructor. In the ThreadedConnectionHandler the Q5 passed is made to equal the Q5 member variable created. This gives a handle to the methods in Q5 without having to create a new Q5/GUI/Graph app for every Client connection.

Q5 is the main handle to the program. It is where the GUI is created and managed (Graph is drawn on GraphGUI). Its following features are

* It contains buttons for Tem plotting on/off for each Client. Displays a rolling average of temps. Allows the user to change sample rate.
* The Temp On/Off buttons on the GUI are Jbuttons which shows usage of swing.
* The GUI is not resizable by the user.
* The GUI includes a max temperature feature if that shows only when the CPU goes above 90 degrees. The max temp feature is my own feature.
* The action event is used to control the inputs from the user to the program (sets temp plotting on/off & allows sample rate to be changed)
* IntTextField class has been overwritten. This prevent user from entering a non-numeric value to the GUI. It is not outlined in the the image \*\*\*

### ThreadedConnectionHandler

The ThreadedConnectionHandler handles the information between the received object and the Q5. It interoperates the objects by using a dummy EmptyClass to get/set the appropriate data. The class is then sent to Q5 when it is made to equal local variables and then sent to GraphGUI for plotting.   
After data is sent successfully to Q5 then the TCH reads if the sample rate has been changed by the user and sends the new sample rate to the Client/EmptyClass pi. The Sample rate is a int used by a thread on both the Client and Server (it is not representative of seconds but is scalable).

### GraphGUI

GraphGUI extends a canvas which is used for plotting temp & time received by the pi. It features the following

* Each plot is represented by a different coloured line.
* The coordinates for the graph are stored in two sets of 5x20 vectors.
* When repaint is called a Boolean bit that was set on by Q5 (depending on the argument passed by the user) is used to determine which Client source is being plotted.
* If the vector becomes longer than 20 the last number of the vector is disregarded.
* The graph displays a line across the screen for a max temp achieved.
* If the temperature goes above 90 then a warning light is turned on for the user to see on the GUI.
* A rolling average for each temp is gathered from the data currently held in the arrays. It is then shown on the GUI for the user to see.
* The graph shows max and min temp for all of the sources plotted across the graph.
* The graph uses an interface to allow data to be passed back to the Q5 class (EE402Increment).
* The GUI contains its own run method from a thread used for testing.
* A for loop is used to paint each line. The lines need to be offseted to the origin of the graph to be useable.

### EmptyClass (PC)

The EmptyClass is a class created to receive and interoperate the objects received from the client. It uses the same set of string/ints in the constructor to create the class that the EmptyClass on the Pi uses. Therefore it is able to interoperate the data. It uses a series of get/set methods to work the relevant data in the THC/Q5 classes.

### EmptyClass (PI)

The EmptyClass Pi is used to generate the data used on the GUI. It includes features.

* A getTempString method is made to that creates a BufferedReader which reads the file containing the CPU temp, this is then assigned to a string and that string is returned. The temp is also displayed on the terminal as the the brief.
* A time method is created that returns the time as a string. It does this by using a Calendar.getInstance() and SimpleDateFormat to get the time and reduce it to a ss format. This means seconds past the last minute change. This time is eventually used as the x variable in the graph. This time is outputted on terminal as per the brief.
* A sample number is increased every time a constructor is called. This sample number is then sent to the Graph.
* A sample rate int is set as a default of 1, which is changeable by the user in the GUI.
* A connection ID string which is set to the input of the user. Note that the user must input a known connection ID (run1-run5).

### Client

The server contains the main run function. It includes the features

* When the correct IP/Connection ID are inputted the run function loops.
* While looping it creates a new EmptyClass, calls the necessary get methods. Then the EmptyClass is sent down object stream.
* Then a thread invokes a sleep for half of the sample rate to allow the data to send and changes to be made on the GUI if needed.
* Then the Client attempts to receive an object. After the receive the tread is invoked again for the reaming half of the sample rate.
* Breaking the thread over 2 periods seems to help the stability of the program as it allows time between the send and receive within the run loop.
* The Client does not flash when ending a message, the code has not been implemented. This is because it was over written when downloading the custom classes from git

## Running Program

When running the clients the program plots the temps successfully. Some things of note are

* The time resets every minute which affects the plotting of the graph.
* There seems to be some spill over between the supplies when plotting multiple temps.
* It takes some time for the graph to become readable.
* It takes a fair amount of heat to show a difference between the min and max temps in the graph.
* Temp On/Off buttons working.
* Temp On/Off freezes current average temp.
* Sample ID is attempted to be displayed beside the last point added to the graph for each source (not very visible)
* Close button ends program.

Figure 4 shows the GUI running S3-S5.

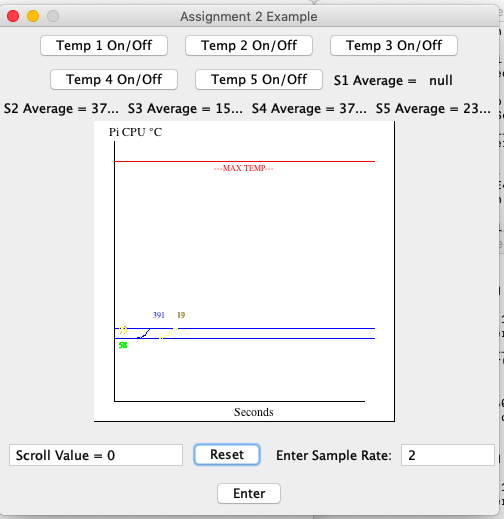


Figure 4

Figure 5 shows the the Q5 terminal, the GUI, along with two Clients running from the Pi.   
In this case S4 and S5 are turned on.

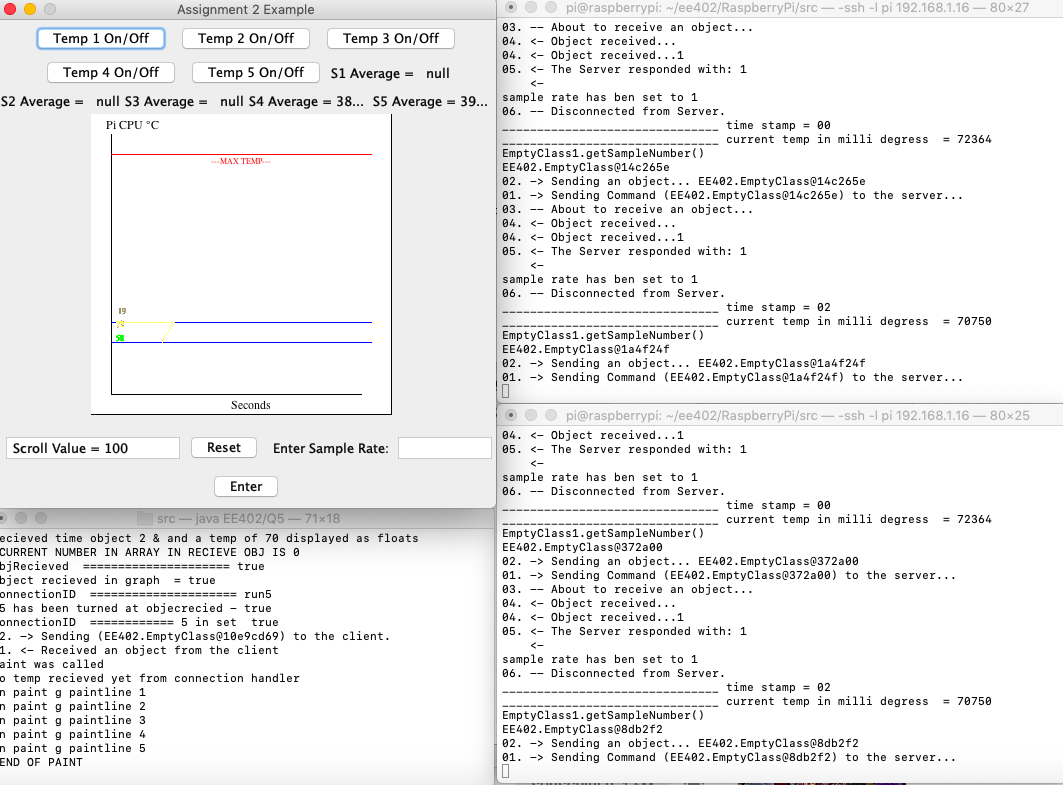


Figure 5

Scroll value was originally intended to allow a scroll of the graph. When it became apparent that the time x value restated every minute this was dropped from the design. Figure 6 shows the terminal message confirming differentiation between sources is made. Many more screenshots of the running program and its deployment have been included in the screenshot folder accompanying this report

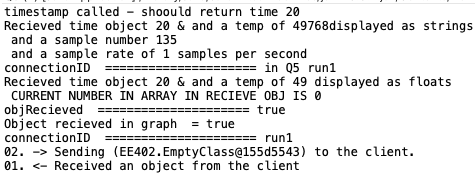


Figure 6

The figure 7 shows all clients reading at the same time.

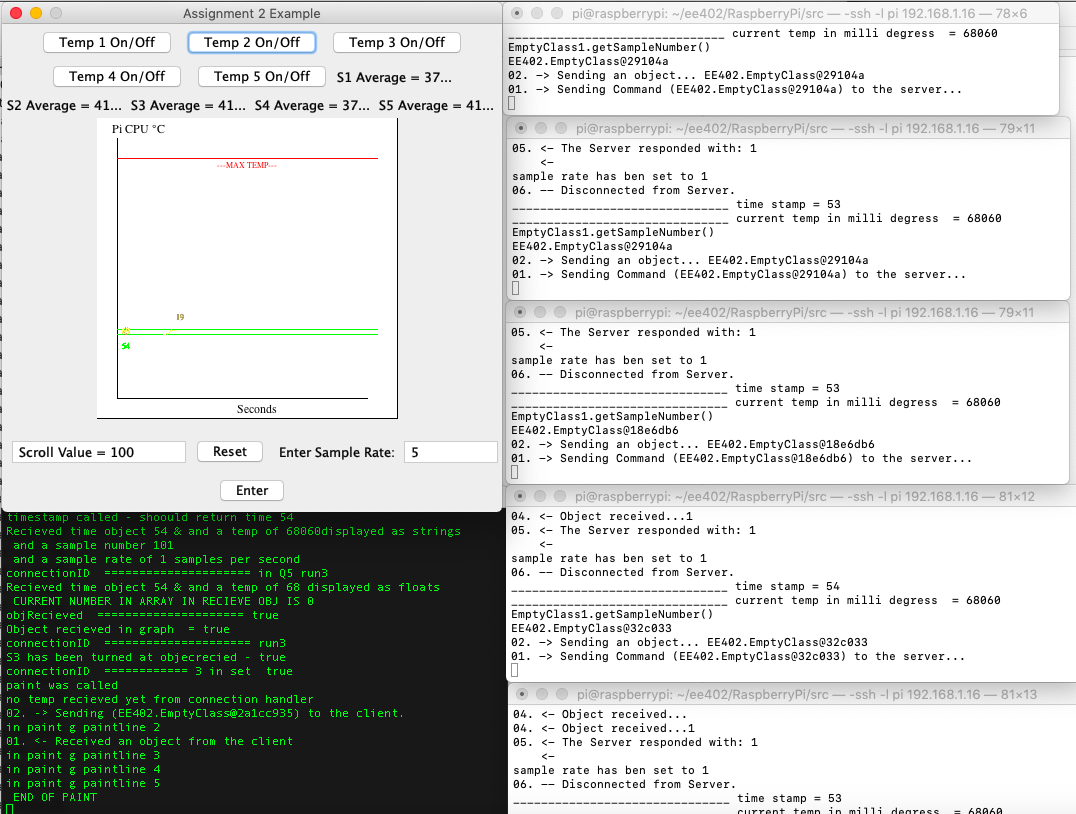


Figure 7

### How to run.

Server and GGUI on PC – See Figure 7

Compiling: When in folder ‘src’ input command javac EE402/\*.java in terminal

Running: From the same ‘src’ folder java EE402/Q5

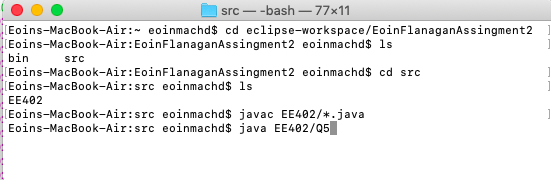


Figure 8

Client on PI – see Figure 9

Navigate to folder ‘src’ as above

Compiling: When in folder ‘src’ input command javac EE402/\*.java in terminal

Running: From the same ‘src’ folder java EE402/Client (IP Address of Server) run1  
Note that run1 can also be run2-5.



Figure 9

# Conclusion

The project proved to be a very labour intensive one. Many hours where spent just setting up the communications to and from the Raspberry Pi. The Raspberry Pi is slow to compile and is not as user friendly as Eclipse when coding. Using a localhost was not known until two weeks before the project due date, significant progress may have been made if this was already known.

Given more time the the graph would be fine tweaked to be more readable. The LED on the Pi would be implemented. There are a few variables that were used during the development of the project that have been left in. This is due to fear of breaking the current progress made close to the deadline, given more time these variables would be deleted as necessary. A few classes are included in the project that no longer used. Q5 is not a good name for a class. Q5 class was built from a class used to answer tutorial 5. An attempt to resolve this issue was made but was proving too time consuming.

That said it provided a great insight into object oriented client/server applications and communicating to embedded Linux. The brief was easy to understand but proved challenging and many new skills have been learned due to the projects scope.

# Source Code

## Q5 Class

package EE402;

import java.awt.\*;

import java.awt.event.\*;

import java.io.Serializable;

import javax.swing.\*;

@SuppressWarnings("serial")

public class Q5 extends JFrame implements Serializable , AdjustmentListener, ActionListener, GGUIServerInterface, EE402Increment, EmptyClassInterface, Runnable

{

private static int y;

private static int x;

private int sampleNumber = 0;

private static int sampleRateInt =1;

private static volatile boolean running = true;

private static volatile boolean on = false;

private static volatile boolean objRecieved;

private int currentNumberInArray;

private static int firstRound =0;

private static boolean fistLoopDone;

private volatile boolean paintLine1 = false; private volatile boolean paintLine2 = false;

private volatile boolean paintLine3 = false; private volatile boolean paintLine4 = false;

private volatile boolean paintLine5 = false;

private boolean maxTemp;

private Color offRed;

private Scrollbar scrollbar;

private IntTextField status;

private Button resetButton;

private GraphGUI GGUI;

// private GraphGUI GGUI2;

private JButton s1State; private boolean temp1bit = true;

private JButton s2State; private boolean temp2bit = true;

private JButton s3State; private boolean temp3bit = true;

private JButton s4State; private boolean temp4bit = true;

private JButton s5State; private boolean temp5bit = true;

private Label sampleRateLbl;

private IntTextField sampleRateTxt;

private String tempReading;

private Button sampleButton;

private int sampleRateInt2 =1;

private Label s1AverageLbl;

private Label s2AverageLbl;

private Label s3AverageLbl;

private Label s4AverageLbl;

private Label s5AverageLbl;

private Label maxTemplbl;

private String s1Average;

private String s2Average;

private String s3Average;

private String s4Average;

private String s5Average;

private EmptyClassInterface ECIapp;

private String temp;

private EmptyClass EmptyClass;

private String time;

private String connectionID;

//private int on;

public Q5(){

super("Assignment 2 Example");

this.setLayout(new FlowLayout());

Panel topPanel = new Panel();

this.setDefaultCloseOperation(this.EXIT\_ON\_CLOSE);

// if (setDefaultCloseOperation = true) {setOnOff

this.status = new IntTextField(20);

this.scrollbar = new Scrollbar(Scrollbar.HORIZONTAL, 100, 10, 0, 110);

this.resetButton = new Button("Reset");

offRed = new Color(128, 0, 0);

this.maxTemplbl = new Label("MAX TEMP REACHED");

maxTemplbl.setBackground(offRed);

this.s1State = new JButton("Temp 1 On/Off");

this.s1State.addActionListener(this);

this.s2State = new JButton("Temp 2 On/Off");

this.s2State.addActionListener(this);

this.s3State = new JButton("Temp 3 On/Off");

this.s3State.addActionListener(this);

this.s4State = new JButton("Temp 4 On/Off");

this.s4State.addActionListener(this);

this.s5State = new JButton("Temp 5 On/Off");

this.s5State.addActionListener(this);

this.getContentPane().add(s1State);

this.getContentPane().add(s2State);

this.getContentPane().add(s3State);

this.getContentPane().add(s4State);

this.getContentPane().add(s5State);

this.s1AverageLbl = new Label("S1 Average = " + " " + s1Average );

this.s2AverageLbl = new Label("S2 Average = " + " " + s2Average );

this.s3AverageLbl = new Label("S3 Average = " + " " + s3Average);

this.s4AverageLbl = new Label("S4 Average = " + " " + s4Average );

this.s5AverageLbl = new Label("S5 Average = " + " " + s5Average );

this.add(s1AverageLbl);

this.add(s2AverageLbl);

this.add(s3AverageLbl);

this.add(s4AverageLbl);

this.add(s5AverageLbl);

this.GGUI = new GraphGUI(this);

this.add(GGUI, BorderLayout.CENTER);

this.sampleRateTxt = new IntTextField(10);

this.sampleRateLbl = new Label("Enter Sample Rate: ");

this.sampleButton = new Button("Enter");

this.sampleButton.addActionListener(this);

this.resetButton.addActionListener(this);

this.scrollbar.addAdjustmentListener(this);

topPanel.add(status);

topPanel.add(resetButton);

this.add(topPanel);

//this.add(scrollbar);

this.add(sampleRateLbl);

this.add(sampleRateTxt);

this.add(sampleButton);

// sampleRate = Integer.valueOf(sampleRateTxt.getSelectedText());

// System.out.println(sampleRate);

if (maxTemp) {

this.add(maxTemplbl);

}

this.update();

this.pack();

this.setBounds(500, 500, 500, 510);

this.setResizable(false);

this.setVisible(true);

new ThreadedServer(this);

}

public void increment() {

this.scrollbar.setValue(scrollbar.getValue()+10);

this.update();

}

public void actionPerformed(ActionEvent e){

if (e.getActionCommand().equals("Reset")){

this.scrollbar.setValue(0);

this.update();

}

if (e.getActionCommand().equals("Enter")){

String sampleRate = this.sampleRateTxt.getText();

// @SuppressWarnings("deprecation")

this.sampleRateInt = Integer.valueOf(sampleRate);

this.sampleRateInt2 = sampleRateInt;

this.setSampleRateQ5(sampleRateInt);

}

if (e.getActionCommand().equals("Temp 1 On/Off")){

if(temp1bit == true) {temp1bit = false; GGUI.setPaintLine1(temp1bit); }

else if (temp1bit == false) {temp1bit = true; GGUI.setPaintLine1(temp1bit);}

System.out.println("S1 has been turned - " + temp1bit);

this.update();

}

if (e.getActionCommand().equals("Temp 2 On/Off")){

if(temp2bit == true) {temp2bit = false; GGUI.setPaintLine1(temp2bit); }

else if (temp2bit == false) {temp2bit = true; GGUI.setPaintLine2(temp2bit);}

System.out.println("S2 has been turned - " + temp2bit);

this.update();

}

if (e.getActionCommand().equals("Temp 3 On/Off")){

if(temp3bit == true) {temp3bit = false; GGUI.setPaintLine1(temp3bit); }

else if (temp3bit == false) {temp3bit = true; GGUI.setPaintLine3(temp3bit);}

System.out.println("S3 has been turned - " + temp3bit);

this.update();

}

if (e.getActionCommand().equals("Temp 4 On/Off")){

if(temp4bit == true) {temp4bit = false; GGUI.setPaintLine1(temp4bit); }

else if (temp4bit == false) {temp4bit = true; GGUI.setPaintLine4(temp4bit);}

System.out.println("S4 has been turned - " + temp4bit);

this.update();

}

if (e.getActionCommand().equals("Temp 5 On/Off")){

if(temp5bit = true) {temp5bit = false; GGUI.setPaintLine1(temp5bit); }

else if (temp5bit == false) {temp5bit = true; GGUI.setPaintLine4(temp5bit);}

System.out.println("S4 has been turned - " + temp5bit);

this.update();

}

}

public int getSampleRateQ5(int sampleRateInt) {return this.sampleRateInt;}

public void setSampleRateQ5(int sampleRateInt) {

this.sampleRateInt = sampleRateInt;

GGUI.setSampleRate(sampleRateInt);

System.out.println("Q5 set SampleRate is " + sampleRateInt);

}

@Override

public int getSampleRateQ5() {

// TODO Auto-generated method stub

return 0;

}

public void adjustmentValueChanged(AdjustmentEvent e){

if (e.getSource().equals(scrollbar)){

this.GGUI.setGraphGUI(scrollbar.getValue());

this.update();

}

}

public void update(){

status.setText("Scroll Value = " + scrollbar.getValue());

if (maxTemp) {

this.add(maxTemplbl);

}

}

public String recieveTemp(String st) {

tempReading = st;

this.update();

System.out.println(st);

return tempReading;

}

@Override

public String getTempString() {

// TODO Auto-generated method stub

return null;

}

@Override

public String doSomthing(String tempString) {

// TODO Auto-generated method stub

return null;

}

// public int setSampleRate(int sampleRateInts) {

// System.out.println("sample rate setting test in Q5 " + sampleRateInts);

// this.sampleRateInt = sampleRateInts;

// return this.sampleRateInt;

// }

public int getSampleRate() {

// String sampleRate = this.app.sampleRateTxt.getText();

return sampleRateInt;}

public boolean objRecieved(boolean b) {return objRecieved;}

public boolean setState(boolean b) {return on;}

@Override

public void run() {

System.out.println(" thread runing in Q5 " );

while(on) {

}

// TODO Auto-generated method stub

}

public boolean maxTemp(boolean maxT) { this.maxTemp = maxT;

return maxTemp; }

public float setGraphX(float x) {return this.x;

}

public float setGraphY(float y) {return this.y;}

public int setSampleNumber (int sampleNumber) {return this.sampleNumber;}

public int setSampleRate (int sampleRate) {return this.sampleRateInt;}

//public void recieveObj3(EmptyClass EmptyClass) {

//this.update();

//}

public void recieveObj2(EmptyClass EmptyClass) {

}

public void recieveObj(int sampleNumber, String time, String tempa, int sampleRate, String connectionID2) {

this.sampleNumber = sampleNumber;

this.temp = tempa;

this.time = time;

this.sampleRateInt = sampleRate;

this.connectionID = connectionID2;

objRecieved = true;

System.out.println("Recieved time object " + time + " & and a temp of " + temp + "displayed as strings");

System.out.println(" and a sample number " + this.sampleNumber);

System.out.println(" and a sample rate of " + this.sampleRateInt + " samples per second");

System.out.println("connectionID ===================== in Q5 " + connectionID);

if (connectionID == "run1") {paintLine1 = true; this.GGUI.setPaintLine1( paintLine1);}

// temp values to change slightly, will be removing the math.rabdom when using pi temp

y = (Integer.valueOf(temp));

float y1 = (float) ((y /1000));

y = Math.round(y1);

x = (Integer.valueOf(this.time));

System.out.println("Recieved time object " + x + " & and a temp of " + y + " displayed as floats");

System.out.println(" CURRENT NUMBER IN ARRAY IN RECIEVE OBJ IS " + currentNumberInArray);

System.out.println("objRecieved ===================== " + objRecieved);

objRecieved();

this.update();

}

public void objRecieved(){

if (objRecieved = true){

GGUI.setObjRecieved(objRecieved);

System.out.println("connectionID ===================== " + connectionID);

if (connectionID.equals("run1") & temp1bit) {

this.GGUI.sendConnectionID(connectionID);

System.out.println("S1 has been turned at objecrecied - " + temp1bit);

paintLine1 = true; this.GGUI.setPaintLine1( paintLine1);

this.GGUI.setTempGraph(y);

this.GGUI.setTimeGraph(x);

this.GGUI.setSampleNuber(sampleNumber);

s1Average = ("S1 Average = " + String.valueOf(GGUI.displayAve1()));

s1AverageLbl.setText(s1Average);

System.out.println("os1Averaged " + s1Average);

this.update();

if (maxTemp) {

this.add(maxTemplbl);

}

}

if (connectionID.equals("run2") & temp2bit) {

this.GGUI.sendConnectionID(connectionID);

System.out.println("S2 has been turned at objecrecied - " + temp2bit);

paintLine2 = true; this.GGUI.setPaintLine2( paintLine2);

this.GGUI.setTempGraph(y);

this.GGUI.setTimeGraph(x);

this.GGUI.setSampleNuber(sampleNumber);

s2Average = ("S2 Average = " + String.valueOf(GGUI.displayAve2()));

s2AverageLbl.setText(s2Average);

this.update();

if (maxTemp) {

this.add(maxTemplbl);

}

//ect for 5

}

if (connectionID.equals("run3") & temp3bit) {

this.GGUI.sendConnectionID(connectionID);

System.out.println("S3 has been turned at objecrecied - " + temp3bit);

paintLine3 = true; this.GGUI.setPaintLine3( paintLine3);

this.GGUI.setTempGraph(y);

this.GGUI.setTimeGraph(x);

this.GGUI.setSampleNuber(sampleNumber);

s3Average = ("S3 Average = " + String.valueOf(GGUI.displayAve3()));

s3AverageLbl.setText(s3Average);

if (maxTemp) {

this.add(maxTemplbl);

}

this.update();

}

if (connectionID.equals("run4") & temp4bit) {

this.GGUI.sendConnectionID(connectionID);

System.out.println("S4 has been turned at objecrecied - " + temp4bit);

paintLine4 = true; this.GGUI.setPaintLine4( paintLine4);

this.GGUI.setTempGraph(y);

this.GGUI.setTimeGraph(x);

this.GGUI.setSampleNuber(sampleNumber);

s4Average = ("S4 Average = " + String.valueOf(GGUI.displayAve4()));

s4AverageLbl.setText(s4Average);

if (maxTemp) {

this.add(maxTemplbl);

}

this.update();

}

if (connectionID.equals("run5") & temp5bit) {

this.GGUI.sendConnectionID(connectionID);

System.out.println("S5 has been turned at objecrecied - " + temp5bit);

paintLine5 = true; this.GGUI.setPaintLine5( paintLine5);

this.GGUI.setTempGraph(y);

this.GGUI.setTimeGraph(x);

this.GGUI.setSampleNuber(sampleNumber);

s5Average = ("S5 Average = " + String.valueOf(GGUI.displayAve5()));

s5AverageLbl.setText(s5Average);

if (maxTemp) {

this.add(maxTemplbl);

}

this.update();

}

}

}

public static void main(String args[]) {

new Q5();

}

public String getConnectionID() {

// TODO Auto-generated method stub

return connectionID;

}

}

## GraphGUI

package EE402;

import java.awt.\*;

import java.awt.event.\*;

import java.io.Serializable;

import java.sql.Date;

import java.util.Timer;

import java.util.Arrays;

import java.util.Collections;

import javax.print.DocFlavor.STRING;

import java.util.\*;

import java.util.Calendar;

public class GraphGUI extends Canvas implements MouseListener, AdjustmentListener, ActionListener, Serializable, Runnable {

private static final long serialVersionUID = 1L;

// private int x;

private EE402Increment app;

private EmptyClassInterface ECIapp;

private float [] y = new float[5];

private float [] x = new float [5];

private int [] sampleNumber = new int [5];

private int sampleRate;

private static int sampleRateInt =1;

private static volatile boolean running = true;

private static volatile boolean on = false;

private static volatile boolean objRecieved;

private static int firstRound =0;

private static boolean fistLoopDone;

private Graphics g;

private Thread thread;

private boolean firstbit;

GraphGUI GraphGUI;

private long threadId;

private long tempthreadId;

private int init =0;

// private int max1; private int min1;

// private int max2; private int min2;

// private int max3; private int min3;

// private int max4; private int min4;

// private int max5; private int min5;

private boolean paintLine1 = false; private boolean paintLine2 = false;

private boolean paintLine3 = false; private boolean paintLine4 = false;

private boolean paintLine5 = false;

private Vector<Integer> vTemp1 = new Vector<Integer>(20); private Vector<Integer> vTime1 = new Vector<Integer>(20);

private Vector<Integer> vTemp2 = new Vector<Integer>(20); private Vector<Integer> vTime2 = new Vector<Integer>(20);

private Vector<Integer> vTemp3 = new Vector<Integer>(20); private Vector<Integer> vTime3 = new Vector<Integer>(20);

private Vector<Integer> vTemp4 = new Vector<Integer>(20); private Vector<Integer> vTime4 = new Vector<Integer>(20);

private Vector<Integer> vTemp5 = new Vector<Integer>(20); private Vector<Integer> vTime5 = new Vector<Integer>(20);

private double [] data;

private int [] [] XData = new int [5][20];

private int [] [] YData = new int [5][20];

private int [] XDataOld = new int [20];

private int [] [] YDataOld = new int [5][20];

private int [] xAcc = new int [5];

private int previousNumberInArray;

private int xOriginOffset = 20;

private int yOriginOffset = 280;

private static int test1, test2, test3, test4 = 250;

private double Xtotal= 0;

private double [] Ytotal = new double [5];

private double [] YAve = new double [5];

private int previousXValue ;

private int previousYValue ;

private static int currentNumberInArray = 0;

private static int currentNumberInArray1 = 0;

private int numberOfThreads = 0;

private int z;

private String temp = "no temp recieved yet";

private String time = "no time recieved yet";

private EmptyClass EmptyClass;

private ThreadedConnectionHandler TCH;

private int setConnectionID;

private String connectionID;

public GraphGUI(EE402Increment app) {

this.setPreferredSize(new Dimension (310,310));

// this.addMouseListener(this);

this.app = app;

this.setVisible(true);

}

public void setPaintLine1(boolean paintLine1) {this.paintLine1 = paintLine1; System.out.println("connectionID ============ 1 in set " + this.paintLine1);}

public void setPaintLine2(boolean paintLine2) {this.paintLine2 = paintLine2; System.out.println("connectionID ============ 2 in set " + this.paintLine2);}

public void setPaintLine3(boolean paintLine3) {this.paintLine3 = paintLine3; System.out.println("connectionID ============ 3 in set " + this.paintLine3);}

public void setPaintLine4(boolean paintLine4) {this.paintLine4 = paintLine4; System.out.println("connectionID ============ 4 in set " + this.paintLine4);}

public void setPaintLine5(boolean paintLine5) {this.paintLine5 = paintLine5; System.out.println("connectionID ============ 5 in set " + this.paintLine5);}

//

public void setGraphGUI(int z) {

this.z = z;

test1 = test1 +1;

test2 = test2 -1;

test3 = test3 +1 ;

test4 = test4 -1;

this.repaint();

}

public void setTempGraph(int y) {

// this.y = y;

if (paintLine1) { this.y[0] = y; }

if (paintLine2) { this.y[1] = y; }

if (paintLine1) { this.vTemp1.add(y); if (vTemp1.size() == 20) { vTemp1.remove(0);}}

if (paintLine2) { this.vTemp2.add(y); if (vTemp2.size() == 20) { vTemp2.remove(0);}}

if (paintLine3) { this.vTemp3.add(y); if (vTemp3.size() == 20) { vTemp3.remove(0);}}

if (paintLine4) { this.vTemp4.add(y); if (vTemp4.size() == 20) { vTemp4.remove(0);}}

if (paintLine5) { this.vTemp5.add(y); if (vTemp5.size() == 20) { vTemp5.remove(0);}}

this.repaint();

}

public void sendConnectionID(String sid) {

this.connectionID = sid;

}

public void setSampleNuber(int x) {

// if (connectionID.equals("run1"))

do {this.sampleNumber[0] = x; break;} while((paintLine1) & connectionID.equals("run1"));

do {this.sampleNumber[1] = x; break;} while((paintLine2) & connectionID.equals("run2"));

do {this.sampleNumber[2] = x; break;} while((paintLine3) & connectionID.equals("run3"));

do {this.sampleNumber[3] = x; break;} while((paintLine4) & connectionID.equals("run4"));

do {this.sampleNumber[4] = x; break;} while((paintLine5) & connectionID.equals("run5"));

this.repaint();

}

public void setTimeGraph(int x){

// this.x = x;

if (paintLine1) { this.x[0] = x; }

if (paintLine2) { this.x[1] = x; }

if (paintLine1) { this.vTime1.add(x); if (vTime1.size() == 20) { vTime1.remove(0);}}

if (paintLine2) { this.vTime2.add(x); if (vTime2.size() == 20) { vTime2.remove(0);}}

if (paintLine3) { this.vTime3.add(x); if (vTime3.size() == 20) { vTime3.remove(0);}}

if (paintLine4) { this.vTime4.add(x); if (vTime4.size() == 20) { vTime4.remove(0);}}

if (paintLine5) { this.vTime5.add(x); if (vTime5.size() == 20) { vTime5.remove(0);}}

this.repaint();

}

public void paint(Graphics g) {

System.out.println("paint was called");

this.g=g;

super.paint(this.g);

System.out.println(temp + " from connection handler");

Font f1 = new Font ("TimesRoman", Font.PLAIN, 12);

this.g.setFont(f1);

this.g.drawRect(0, 0, 300, 300);

this.g.setColor(Color.WHITE);

this.g.fillRect(0, 0, 300, 300);

this.g.setColor(Color.BLACK);

this.g.drawLine(20, 280, 270, 280);

this.g.drawLine(20, 280, 20, 20);

this.g.drawString("Seconds", 140, 295);

this.g.drawString("Pi CPU °C", 15, 15);

this.g.setColor(Color.RED);

this.g.drawLine(20, 40, 280, 40);

Font f2 = new Font ("TimesRoman", Font.PLAIN, 8);

this.g.setFont(f2);

this.g.drawString("---MAX TEMP---", 120 ,50);

this.g.setColor(Color.BLACK);

if(objRecieved = true) {

int max1 = 0; int min1 = 0;

int max2 = 0; int min2 = 0;

int max3 = 0; int min3 = 0;

int max4 = 0; int min4 = 0;

int max5 = 0; int min5 = 0;

//Ploting S1

if(paintLine1) { System.out.println("in paint g paintline 1");

g.setColor(Color.BLUE);

int x2 = 0;int y2 = 0;

for(int i = 1 ; i<vTemp1.size() ; i++) {

int y= this.vTemp1.elementAt(i);

int yPrevious = this.vTemp1.elementAt(i-1);

int x1 = vTime1.elementAt(i-1) + 5;

int y1 = yPrevious;

x2 = this.vTime1.elementAt(i) + 5;

y2 = y ;

g.drawLine(xOriginOffset +x1,yOriginOffset - y1,xOriginOffset + x2, yOriginOffset -y2);

int maxTemp1 = Collections.max(vTemp1);

String SmaxTemp1 = String.valueOf(maxTemp1);

g.drawString(SmaxTemp1, xOriginOffset +5, (yOriginOffset - maxTemp1 +5) );

// g.drawLine(xOriginOffset ,yOriginOffset - maxTemp1, 300 - xOriginOffset , yOriginOffset - maxTemp1);

Ytotal[0] = Ytotal[0] + vTemp1.elementAt(i);

YAve [0] = Ytotal[0] / vTemp1.size();

max1 = Collections.max(vTemp1); min1 = Collections.min(vTemp1);

}

int gs = vTime1.size();

String gs2 = String.valueOf(gs);

g.drawString(gs2,xOriginOffset + x2,yOriginOffset - y2 - 10);

// displaying max and min.. repeated for each of the 5

if(max1 > max2 & max1 > max3 & max1 > max4 & max1 > max5 ) {g.drawLine(xOriginOffset ,yOriginOffset - max1, 300 - xOriginOffset , yOriginOffset - max1); }

if(min1 > min2 & min1 > min3 & min1 > min4 & min1 > min5 ) {g.drawLine(xOriginOffset ,yOriginOffset - min1, 300 - xOriginOffset , yOriginOffset - min1); }

}

//Ploting S2

if(paintLine2) { System.out.println("in paint g paintline 2");

g.setColor(Color.GREEN);

int x2 = 0;int y2 = 0;

for(int i = 1 ; i<vTemp2.size() ; i++) {

int y= this.vTemp2.elementAt(i);

int yPrevious = this.vTemp2.elementAt(i-1);

int x1 = vTime2.elementAt(i-1) + 5;

int y1 = yPrevious;

x2 = this.vTime2.elementAt(i) + 5;

y2 = y ;

g.drawLine(xOriginOffset +x1,yOriginOffset - y1,xOriginOffset + x2, yOriginOffset -y2);

int maxTemp1 = Collections.max(vTime2);

String SmaxTemp1 = String.valueOf(maxTemp1);

g.drawString(SmaxTemp1, xOriginOffset +5, (yOriginOffset - maxTemp1 +5) );

// g.drawLine(xOriginOffset ,yOriginOffset - maxTemp1, 300 - xOriginOffset , yOriginOffset - maxTemp1);

Ytotal [1] = Ytotal[1] + vTemp2.elementAt(i);

YAve [1] = Ytotal[1] / vTemp2.size();

max2 = Collections.max(vTemp2); min2 = Collections.min(vTemp2);

}

int gs = vTime2.size();

String gs2 = String.valueOf(gs);

g.drawString(gs2,xOriginOffset + x2,yOriginOffset - y2 - 10);

if(max2 > max1 & max2 > max3 & max2 > max4 & max2 > max5 ) {g.drawLine(xOriginOffset ,yOriginOffset - max2, 300 - xOriginOffset , yOriginOffset - max2);}

if(min2 > min1 & min2 > min3 & min2 > min4 & min2 > min5 ) {g.drawLine(xOriginOffset ,yOriginOffset - min2, 300 - xOriginOffset , yOriginOffset - min2);}

}

//Ploting S3

if(paintLine3) { System.out.println("in paint g paintline 3");

g.setColor(Color.RED);

int x2 = 0;int y2 = 0;

for(int i = 1 ; i<vTemp3.size() ; i++) {

int y= this.vTemp3.elementAt(i);

int yPrevious = this.vTemp3.elementAt(i-1);

int x1 = vTime3.elementAt(i-1) + 5;

int y1 = yPrevious;

x2 = this.vTime3.elementAt(i) + 5;

y2 = y ;

g.drawLine(xOriginOffset +x1,yOriginOffset - y1,xOriginOffset + x2, yOriginOffset -y2);

int maxTemp1 = Collections.max(vTemp3);

String SmaxTemp1 = String.valueOf(maxTemp1);

g.drawString(SmaxTemp1, xOriginOffset +5, (yOriginOffset - maxTemp1 +5) );

// g.drawLine(xOriginOffset ,yOriginOffset - maxTemp1, 300 - xOriginOffset , yOriginOffset - maxTemp1);

Ytotal[2] = Ytotal[2] + vTemp3.elementAt(i);

YAve [2] = Ytotal[2] / vTemp3.size();

max3 = Collections.max(vTemp3); min3 = Collections.min(vTemp3);

}

int gs = vTime3.size();

String gs2 = String.valueOf(gs);

g.drawString(gs2,xOriginOffset + x2,yOriginOffset - y2 - 10);

if(max3 > max1 & max3 > max2 & max3 > max4 & max3 > max5 ) {g.drawLine(xOriginOffset ,yOriginOffset - max3, 300 - xOriginOffset , yOriginOffset - max3);}

if(min3 > min1 & min3 > min2 & min3 > min4 & min3 > min5 ) {g.drawLine(xOriginOffset ,yOriginOffset - min3, 300 - xOriginOffset , yOriginOffset - min3);}

}

//Ploting S4

if(paintLine4) { System.out.println("in paint g paintline 4");

g.setColor(new Color(255-102-50));

int x2 = 0;int y2 = 0;

for(int i = 1 ; i<vTemp4.size() ; i++) {

int y= this.vTemp4.elementAt(i);

int yPrevious = this.vTemp4.elementAt(i-1);

int x1 = vTime4.elementAt(i-1) + 5;

int y1 = yPrevious;

x2 = this.vTime4.elementAt(i) + 5;

y2 = y ;

g.drawLine(xOriginOffset +x1,yOriginOffset - y1,xOriginOffset + x2, yOriginOffset -y2);

int maxTemp1 = Collections.max(vTemp4);

String SmaxTemp1 = String.valueOf(maxTemp1);

//

g.drawString(SmaxTemp1, xOriginOffset +5, (yOriginOffset - maxTemp1 +5) );

// g.drawLine(xOriginOffset ,yOriginOffset - maxTemp1, 300 - xOriginOffset , yOriginOffset - maxTemp1);

//

Ytotal[3] = Ytotal[3] + vTemp4.elementAt(i);

YAve[3] = Ytotal[3] / vTemp4.size();

max4 = Collections.max(vTemp4); min4 = Collections.min(vTemp4);

}

int gs = vTime4.size();

String gs2 = String.valueOf(gs);

g.drawString(gs2,xOriginOffset + x2,yOriginOffset - y2 - 10);

if(max4 > max1 & max4 > max2 & max4 > max3 & max4 > max5 ) {g.drawLine(xOriginOffset ,yOriginOffset - max4, 300 - xOriginOffset , yOriginOffset - max4);}

if(min4 > min1 & min4 > min2 & min4 > min3 & min4 > min5 ) {g.drawLine(xOriginOffset ,yOriginOffset - min4, 300 - xOriginOffset , yOriginOffset - min4);}

}

//Ploting S4

if(paintLine5) { System.out.println("in paint g paintline 5");

g.setColor(new Color(153-153-153));

int x2 = 0;int y2 = 0;

for(int i = 1 ; i<vTemp5.size() ; i++) {

int y= this.vTemp5.elementAt(i);

int yPrevious = this.vTemp5.elementAt(i-1);

int x1 = vTime5.elementAt(i-1) + 5;

int y1 = yPrevious;

x2 = this.vTime5.elementAt(i) + 5;

y2 = y ;

g.drawLine(xOriginOffset +x1,yOriginOffset - y1,xOriginOffset + x2, yOriginOffset -y2);

int maxTemp1 = Collections.max(vTemp5);

String SmaxTemp1 = String.valueOf(maxTemp1);

g.drawString(SmaxTemp1, xOriginOffset +5, (yOriginOffset - maxTemp1 +5) );

Ytotal[4] = Ytotal[4] + vTemp5.elementAt(i);

YAve[4] = Ytotal[4] / vTemp5.size();

max5 = Collections.max(vTemp5); min5 = Collections.min(vTemp5);

}

int gs = vTime5.size();

String gs2 = String.valueOf(gs);

g.drawString(gs2,xOriginOffset + x2,yOriginOffset - y2 - 10);

if(max5 > max1 & max5 > max2 & max5 > max3 & max5 > max4 ) {g.drawLine(xOriginOffset ,yOriginOffset - max5, 300 - xOriginOffset , yOriginOffset - max5);}

if(min5 > min1 & min5 > min2 & min5 > min3 & min5 > min4 ) {g.drawLine(xOriginOffset ,yOriginOffset - min5, 300 - xOriginOffset , yOriginOffset - min5);}

}

// //max lines

// if(max1 > max2 & max1 > max3 & max1 > max4 & max1 > max5 ) {g.drawLine(xOriginOffset ,yOriginOffset - max1, 300 - xOriginOffset , yOriginOffset - max1); }

//

// if(max2 > max1 & max2 > max3 & max2 > max4 & max2 > max5 ) {g.drawLine(xOriginOffset ,yOriginOffset - max2, 300 - xOriginOffset , yOriginOffset - max2);}

//

// if(max3 > max1 & max3 > max2 & max3 > max4 & max3 > max5 ) {g.drawLine(xOriginOffset ,yOriginOffset - max3, 300 - xOriginOffset , yOriginOffset - max3);}

//

// if(max4 > max1 & max4 > max2 & max4 > max3 & max4 > max5 ) {g.drawLine(xOriginOffset ,yOriginOffset - max4, 300 - xOriginOffset , yOriginOffset - max4);}

//

// if(max5 > max1 & max5 > max2 & max5 > max3 & max5 > max4 ) {g.drawLine(xOriginOffset ,yOriginOffset - max5, 300 - xOriginOffset , yOriginOffset - max5);}

//

//

// this.g.setColor(Color.LIGHT\_GRAY);

// // min lines

// if(min1 > min2 & min1 > min3 & min1 > min4 & min1 > min5 ) {g.drawLine(xOriginOffset ,yOriginOffset - min1, 300 - xOriginOffset , yOriginOffset - min1); }

//

// if(min2 > min1 & min2 > min3 & min2 > min4 & min2 > min5 ) {g.drawLine(xOriginOffset ,yOriginOffset - min2, 300 - xOriginOffset , yOriginOffset - min2);}

//

// if(min3 > min1 & min3 > min2 & min3 > min4 & min3 > min5 ) {g.drawLine(xOriginOffset ,yOriginOffset - min3, 300 - xOriginOffset , yOriginOffset - min3);}

//

// if(min4 > min1 & min4 > min2 & min4 > min3 & min4 > min5 ) {g.drawLine(xOriginOffset ,yOriginOffset - min4, 300 - xOriginOffset , yOriginOffset - min4);}

//

// if(min5 > min1 & min5 > min2 & min5 > min3 & min5 > min4 ) {g.drawLine(xOriginOffset ,yOriginOffset - min5, 300 - xOriginOffset , yOriginOffset - min5);}

//

//

// etc

// this.g.setColor(Color.BLACK); //only doing this for 2 of 5 supplies.......

// if (max1 > max2) {this.g.drawString(String.valueOf(max1), 10, 280 - max1);} else {this.g.drawString(String.valueOf(max2), 10, 280 - max2);}

// if (min1 > min2) {this.g.drawString(String.valueOf(min1), 10, 280 - min1);} else {this.g.drawString(String.valueOf(min1), 10, 280 - min1);}

if ((max1 | max2 | max3 | max4 | max5) > 90 ) { // if any max is over 90 degrees display message

app.maxTemp(true);

}

}

app.update(); //??????

System.out.println(" END OF PAINT");

}

public GraphGUI() {EmptyClass EmptyClass = new EmptyClass(); this.EmptyClass = EmptyClass; }

public boolean setObjRecieved(boolean yn) {

this.objRecieved = yn;

System.out.println("Object recieved in graph = " + objRecieved);

return this.objRecieved;}

public boolean getObjRecieved() {

// System.out.println("GETOBJREC = " + objRecieved);

return objRecieved;}

public boolean getObjRecieved(boolean yn) { yn = objRecieved; return yn;}

public int setSampleRate(int sampleRateInts) {

System.out.println("sample rate setting test in GraphGUI " + sampleRateInts);

this.sampleRateInt = sampleRateInts;

return this.sampleRateInt;

}

public int getSampleRate() {

// String sampleRate = this.app.sampleRateTxt.getText();

return sampleRateInt;}

public double displayAve1() {

double temp0 = YAve[0];

return temp0;

}

public double displayAve2() {

double temp1 = YAve[1];

return temp1;

}

public double displayAve3() {

double temp2 = YAve[2];

return temp2;

}

public double displayAve4() {

double temp3 = YAve[3];

return temp3;

}

public double displayAve5() {

double temp4 = YAve[4];

return temp4;

}

@Override

public void mouseClicked(MouseEvent e) {

System.out.println("The mouse was clicked on the canvas");

this.app.increment();

this.app.update();

// TODO Auto-generated method stub

}

public void update(Graphics g) {

paint(g);

}

@Override

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

}

@Override

public void adjustmentValueChanged(AdjustmentEvent e) {

// TODO Auto-generated method stub

}

@Override

public void run() { on = true; numberOfThreads++;

}

public Thread getThread() {

return thread;

}

public void setThread(Thread thread) {

this.thread = thread;

}

}

## IntTextField

package EE402;

import java.awt.TextField;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

@SuppressWarnings("serial")

public class IntTextField extends TextField implements KeyListener {

public IntTextField(int size){

super(size);

this.addKeyListener(this);

}

public void keyPressed(KeyEvent e) {}

public void keyReleased(KeyEvent e) {}

public void keyTyped(KeyEvent e) {

char c = e.getKeyChar();

if(Character.isDigit(c)||c==KeyEvent.VK\_DELETE||c==KeyEvent.VK\_BACK\_SPACE){

System.out.println("Numeric key pressed");

}

else {

System.out.println("Non numeric key pressed");

e.consume();

}

}

}

## EE402Incrament

**package** EE402;

**public** **interface** EE402Increment {

**public** **void** setSampleRateQ5(**int** sampleRateInt);

**public** **void** increment();

//public void update();

// public String sendSampleRate();

**public** **void** update();

**public** **int** getSampleRateQ5();

**public** String getConnectionID();

**public** **boolean** maxTemp(**boolean** b);

}

## EmptyClass – PC

package EE402;

import java.io.\*;

import java.text.SimpleDateFormat;

import java.net.\*;

public class EmptyClass implements Serializable {

private int sampleNumber;

private String temp;

private String timeStamp = "Server timeStamp";

private int sampleRate = 0;

private String connectionID = "0";

private static final long serialVersionUID = 1L;

// public EmptyClass () {System.out.println("Empty Class Server called");}

public EmptyClass () {}

public EmptyClass (int SampleNumber, String time, String temp, int sampleRate, int connectionID) {}

public String setConnectionID (String s) { connectionID = s; return connectionID;}

public String getConnectionID () {return connectionID;}

public int getSampleRate () {return sampleRate;}

public int setSampleRate(int setSampleRate) {System.out.println("sample rate has ben set to " + setSampleRate); sampleRate = setSampleRate; return sampleRate;}

public Integer getSampleNumber() {return sampleNumber;}

public String getTempString() {return temp;}

public String doSomthing (String tempa) {return this.temp;}

public String time() { System.out.println("timestamp called - shoould return time " + timeStamp);

return timeStamp;}

}

## ThreadedServer

package EE402;

import java.net.\*;

import java.util.Vector;

import java.io.\*;

public class ThreadedServer

{

private static int portNumber = 5050;

private GraphGUI myGraphGUI;

Thread newThread;

private Q5 Q5;

// Vector<Q5> Q51 = new Vector<Q5>(5);

public ThreadedServer(EE402.Q5 Q5) {

// new Q5( );

this.Q5 = Q5;

boolean listening = true;

ServerSocket serverSocket = null;

// Thread newThread = new Thread (new Q5());

// Set up the Server Socket

// this.newThread.start();

try

{

serverSocket = new ServerSocket(portNumber);

System.out.println("New Server has started listening on port: " + portNumber );

}

catch (IOException e)

{

System.out.println("Cannot listen on port: " + portNumber + ", Exception: " + e);

System.exit(1);

}

// Server is now listening for connections or would not get to this point

while (listening) // almost infinite loop - loop once for each client request

{

Socket clientSocket = null;

try{

System.out.println("\*\*. Listening for a connection...");

clientSocket = serverSocket.accept();

System.out.println("00. <- Accepted socket connection from a client: ");

System.out.println(" <- with address: " + clientSocket.getInetAddress().toString());

System.out.println(" <- and port number: " + clientSocket.getPort());

}

catch (IOException e){

System.out.println("XX. Accept failed: " + portNumber + e);

listening = false; // end the loop - stop listening for further client requests

}

// this.Q5 = new Q5();

ThreadedConnectionHandler con = new ThreadedConnectionHandler(clientSocket, this.Q5);

con.start();

System.out.println("02. -- Finished communicating with client:" + clientSocket.getInetAddress().toString());

}

// Server is no longer listening for client connections - time to shut down.

try

{

System.out.println("04. -- Closing down the server socket gracefully.");

serverSocket.close();

}

catch (IOException e)

{

System.err.println("XX. Could not close server socket. " + e.getMessage());

}

}

}

## ThreadedConnectionHandler

package EE402;

import java.net.\*;

import java.util.\*;

import java.io.\*;

public class ThreadedConnectionHandler extends Thread implements Serializable, Runnable

{

/\*\*

\*/

private static final long serialVersionUID = 1L;

private static final Vector<EE402.Q5> Vector = null;

private Socket clientSocket = null; // Client socket object

private ObjectInputStream is = null; // Input stream

private ObjectOutputStream os = null; // Output stream

private DateTimeService theDateService;

private GraphGUI GraphGUI;

private Q5 Q5;

private EmptyClass EmptyClass = new EmptyClass();

private Thread ThreadGUI;

private Thread threadGraph;

private long connectionID;

private Q5toCH app;

private int numbI=0;

private boolean running = true, paused = false;

private Thread thread;

private volatile boolean objectRecieved;

// The constructor for the connection handler

public ThreadedConnectionHandler(Socket clientSocket, EE402.Q5 Q5) {

this.clientSocket = clientSocket;

//Set up a service object to get the current date and time

theDateService = new DateTimeService();

this.Q5 = Q5;

threadGraph = new Thread (this.GraphGUI);

threadGraph.start();

}

// Will eventually be the thread execution method - can't pass the exception back

public void run() {

// for(;numbI <= 5;) {

try {

this.is = new ObjectInputStream(clientSocket.getInputStream());

this.os = new ObjectOutputStream(clientSocket.getOutputStream());

// this.GraphGUI = new GraphGUI();

// this.Q5 = super Q5();

// this.Q5 = new Q5();

while (this.readCommand()) {}

}

catch (IOException e)

{

System.out.println("XX. There was a problem with the Input/Output Communication:");

e.printStackTrace();

}

// }

}

// Receive and process incoming string commands from client socket

private boolean readCommand() {

this.objectRecieved = false;

Q5.setState(false);

Q5.objRecieved(false);

try {

EmptyClass = (EmptyClass) is.readObject();

System.out.println(" recieveing object " + EmptyClass.getSampleRate() + ".");

System.out.println("Temp is " + EmptyClass.getTempString());

}

catch (Exception e){

this.closeSocket();

System.out.println("About to close sockets");

return false;

}

Q5.recieveObj(EmptyClass.getSampleNumber(), EmptyClass.time(), EmptyClass.getTempString(), EmptyClass.getSampleRate(), EmptyClass.getConnectionID());

// Q5.recieveObj3(EmptyClass);

Q5.objRecieved(false);

int a = Q5.getSampleRate();

int b = EmptyClass.getSampleRate();

if (a != b)

{

EmptyClass.setSampleRate(a);

System.out.println("Sample rate has been set to " + EmptyClass.getSampleRate());

}

Q5.objRecieved(false);

send(EmptyClass);

System.out.println("01. <- Received an object from the client");// (" + EmptyClass.toString() + " - " + EmptyClass.getTempString() + ").");

try {

int x = Q5.getSampleRate();

Thread.sleep((0));

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return true;

}

public void sendObj(int SampleNumber, String Time2, String temp, int sampleRate) {

}

// Use our custom DateTimeService Class to get the date and time

private void getDate() { // use the date service to get the date

String currentDateTimeText = theDateService.getDateAndTime();

this.send(currentDateTimeText);

}

// Send a generic object back to the client

private void send(Object o) {

try {

System.out.println("02. -> Sending (" + o +") to the client.");

this.os.writeObject(o);

this.os.flush();

}

catch (Exception e) {

System.out.println("XX." + e.getStackTrace());

}

}

// Send a pre-formatted error message to the client

public void sendError(String message) {

this.send("Error:" + message); //remember a String IS-A Object!

}

// Close the client socket

public void closeSocket() { //gracefully close the socket connection

try {

this.os.close();

this.is.close();

this.clientSocket.close();

}

catch (Exception e) {

System.out.println("XX. " + e.getStackTrace());

}

}

}

## Client

package EE402;

//import ee402.\*;

/\* The Client Class - Written by Derek Molloy for the EE402 Module

\* See: ee402.eeng.dcu.ie

\*

\*

\*/

import java.net.\*;

import java.io.\*;

public class Client extends Thread implements TempClientInterface, Serializable, EmptyClassServerInterface{

/\*\*

\*

\*/

private static final long serialVersionUID = 1L;

private static int portNumber = 5050;

private Socket socket = null;

private ObjectOutputStream os = null;

private ObjectInputStream is = null;

private String temp;

static Client Clientapp;

private boolean running = true, paused = false;

private Thread thread;

private EmptyClass EmptyClass1;

private static String connectionID;

private int y;

// the constructor expects the IP address of the server - the port is fixed

public Client(String serverIP, String Temp1of5) {

if (!connectToServer(serverIP)) {

System.out.println("XX. Failed to open socket connection to: " + serverIP);

}

//this.Clientapp = (this);

this.thread = new Thread(this);

//

}

private boolean connectToServer(String serverIP) {

try { // open a new socket to the server

this.socket = new Socket(serverIP,portNumber);

this.os = new ObjectOutputStream(this.socket.getOutputStream());

this.is = new ObjectInputStream(this.socket.getInputStream());

System.out.println("00. -> Connected to Server:" + this.socket.getInetAddress()

+ " on port: " + this.socket.getPort());

System.out.println(" -> from local address: " + this.socket.getLocalAddress()

+ " and port: " + this.socket.getLocalPort());

}

catch (Exception e) {

System.out.println("XX. Failed to Connect to the Server at port: " + portNumber);

System.out.println(" Exception: " + e.toString());

return false;

}

return true;

}

private void RunApp() {

int sn = 0;

EmptyClass EmptyClass1 = new EmptyClass();

this.EmptyClass1 = new EmptyClass();

while(running) {

int orginalSampleRate = EmptyClass1.getSampleRate();

y = EmptyClass1.getSampleRate();

EmptyClass1.getSampleNumber(); EmptyClass1.time(); EmptyClass1.getTempString(); EmptyClass1.getSampleRate();EmptyClass1.setConnectionID(connectionID);

System.out.println("EmptyClass1.getSampleNumber()");

send(EmptyClass1);

System.out.println("01. -> Sending Command (" + EmptyClass1 + ") to the server...");

try {

Thread.sleep(y\*2 \* 1000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

try{

EmptyClass1 = receive();

System.out.println("05. <- The Server responded with: " + EmptyClass1.getSampleRate());

System.out.println(" <- " );

EmptyClass1.setSampleRate(EmptyClass1.getSampleRate());

}

catch (Exception e){

System.out.println("XX. There was an invalid object sent back from the server");

}

System.out.println("06. -- Disconnected from Server.");

}

try {

Thread.sleep(y\* 2 \* 1000);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

public void stopCount() { this.running = false; }

// method to send a generic object.

private void send(Object o) {

// o = ( "" + o);

try {

System.out.println(o);

System.out.println("02. -> Sending an object... " + o);

os.writeObject(o);

os.flush();

}

catch (Exception e) {

System.out.println("XX. Exception Occurred on Sending:" + o + " -- " + e.toString());

this.stopCount();

// System.out.println(o);

}

}

// method to receive a generic object.

private EmptyClass receive()

{

EmptyClass o = null;

EmptyClass EmptyClass = new EmptyClass();

try {

System.out.println("03. -- About to receive an object...");

// o = (EmptyClass) is.readObject();

EmptyClass = (EE402.EmptyClass) is.readObject();

System.out.println("04. <- Object received..." );

System.out.println("04. <- Object received..." + EmptyClass.getSampleRate());

}

catch (Exception e) {

System.out.println("XX. Exception Occurred on Receiving:" + e.toString());

}

return EmptyClass;

}

public static void main(String args[])

{

System.out.println("\*\*. Java Client Application - EE402 OOP Module, DCU");

if(args.length==2){

Client theApp = new Client(args[0], args[1]);

connectionID=args[1];

theApp.RunApp();

}

else

{

System.out.println("Error: you must provide the address of the server");

System.out.println("Usage is: java Client x.x.x.x (e.g. java Client 192.168.7.2)");

System.out.println(" or: java Client hostname (e.g. java Client localhost)");

}

System.out.println("\*\*. End of Application.");

}

@Override

public String getTempString() {

// TODO Auto-generated method stub

return temp;

}

}

## EmptyClass (Pi)

package EE402;

import java.io.\*;

import java.net.\*;

import java.util.Calendar;

import java.text.SimpleDateFormat;

public class EmptyClass implements Serializable {

private int sampleNumber = 0;

private String temp = "default";

private String tempString = "tempString";

private String timeStamp = "Server time....";

private int sampleRate = 1; // default of 1 per second

private String connectionID;

private static final long serialVersionUID = 1L;

Client Clientapp;

public EmptyClass() {this.sampleNumber = sampleNumber;}

public EmptyClass(Client Clientapp) {this.Clientapp = Clientapp;}

public EmptyClass (int sampleRate) { this.sampleRate = sampleRate;}

public int getSampleNumber() {return sampleNumber++;}

public int setSampleRate(int setSampleRate) {System.out.println("sample rate has ben set to " + setSampleRate); sampleRate = setSampleRate; return sampleRate;}

public String setConnectionID (String s) { connectionID = s; return connectionID;}

public EmptyClass (String temp) {

this.temp = temp;

System.out.println("Empty Class Pi String called");}

public EmptyClass(String time, String temp) {

this.temp = temp;

this.timeStamp = time;

}

public EmptyClass (int sampleNumber, String time, String temp, int sampleRate, String connectionID) {

this.sampleNumber = sampleNumber;

this.temp = temp;

this.timeStamp = time;

this.sampleRate = sampleRate;

this.connectionID = connectionID;

}

//public int getSampleRate () {return sampleRate;}

public int getSampleRate () {return sampleRate;}

public String time(){

Calendar cal = Calendar.getInstance();

SimpleDateFormat sdf = new SimpleDateFormat("ss");

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ time stamp = " + sdf.format(cal.getTime()));

// timeStamp = sdf.format(cal.getTime()).toString();

timeStamp = ((sdf.format(cal.getTime()).toString()));

return timeStamp;

}

public String getTempString() { // get temp

BufferedReader in = null;

try {

//in = new BufferedReader(new FileReader("/Users/eoinmachd/Desktop/FakeTempRead.txt"));

in = new BufferedReader(new FileReader("/sys/class/thermal/thermal\_zone0/temp"));

}

catch (FileNotFoundException e) {e.printStackTrace();}

try {tempString = in.readLine();}

catch (IOException e) {e.printStackTrace();}

temp = tempString;

System.out.println("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ current temp in milli degress = " + temp);

return temp;}

}