Java Project

Léon PASSERA, Mathis POTEAU

Problem: Green-Er consumption application

ASI

2023-2024

Contents

1 General Obejctive

Sensors in the GreenEr building measure its energy consumption and production thanks to the photovoltaic panels installed on the roof. A weather station measures meteorological data such as temperature and solar radiation. The corresponding measurements covering the period September, 1st 2022 to August, 31th 2023 are given in the file named 'Green_Er_Energy_Consumption.csv'. Data have been sampled with a 1 hour time resolution. The comma-separated values text file format has been chosen because it is easy to read and widespread. The objective here is to create an application which displays the measurements from that period.

2 Code Layout

The `Main` class of the program call the class `FenPrincipale` which aims to display a window that allows the user to choose the viewing dates and which data he desires to see. These data was retrieve from `'Green_Er_Energy_Consumption.csv'` thanks to the class `DataContainer` with the variable `recupDonnées`.

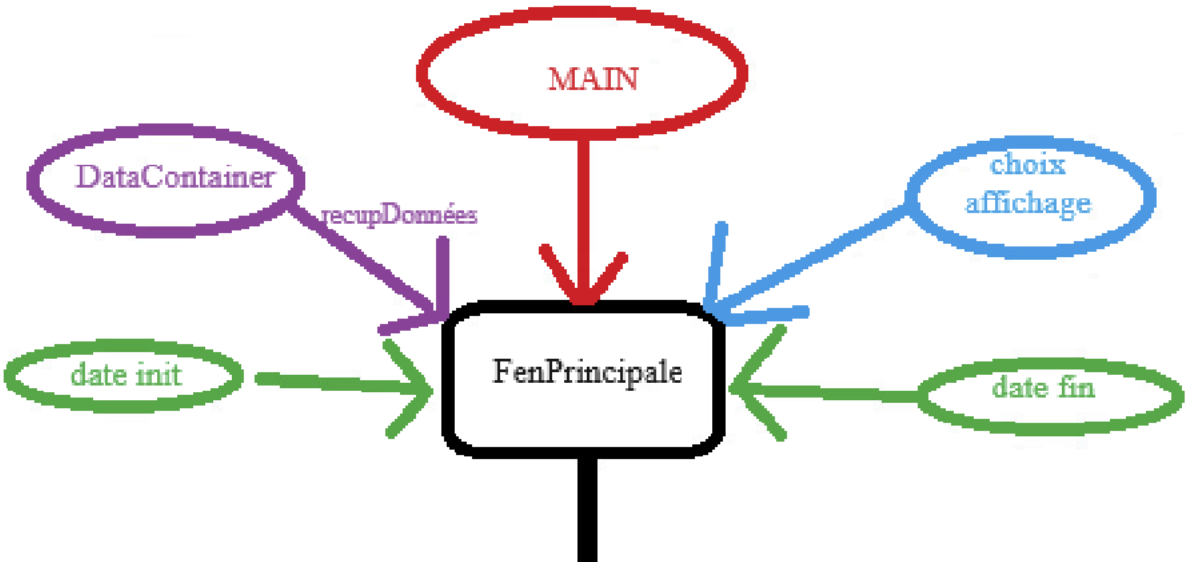

Figure 1 – Code diagram from `Main` to `FenPrincipale`

Then, `FenPrincipale` calls the class `ChoixDateBouton` that check if the dates entered by the user are between the allowed period.

- If the dates are not between the period, `ChoixDateBouton` opens an error window that can only be closed by a button piloted by the class `okErreurBouton`.
- If the dates are all good, `ChoixDateBouton` calls the class `PlotGraph` that will plot the wanted data during the desired period.

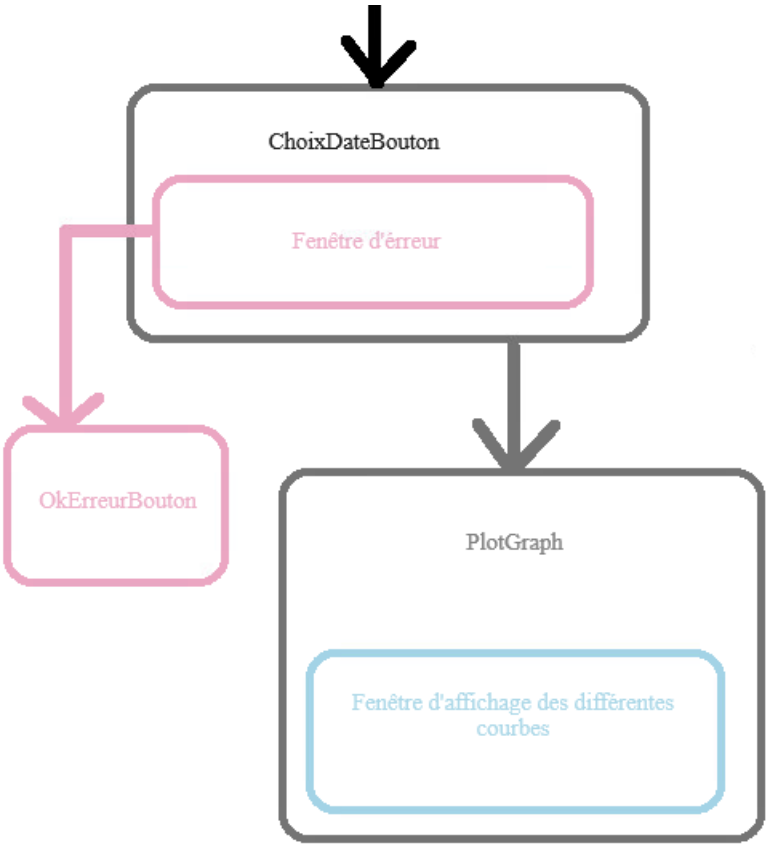

Figure 2 – Code diagram from `ChoixDateBouton` to `PlotGraph`

2.1 The class `FenPincipale`

`FenPincipale` diplays the main window where the user has to choose wich measurement variables he wants to plot the period he wants to see. The first and last dates from this period has to be between `2022-09-01 23:00:00` and `2023-08-31 00:00:00`. The use can display 5 measurements variables :

- `Gree_Er_Consumption_kW`
- `Gree_Er_Production_kW`
- `Outdoor Temperature`
- `Global Radiation`
- `Gree_Er_Consumption_kW - Gree_Er_Production_kW`

In addtion, the user can choose if he wants to display his graph hourly, daily or monthly.

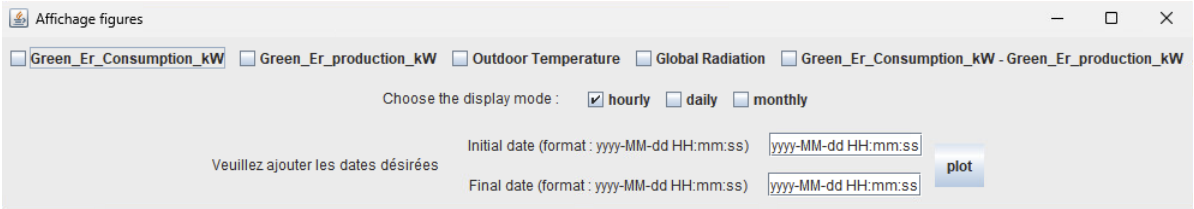

Figure 3 – Main window

2.2 The class `ChoixDateBouton`

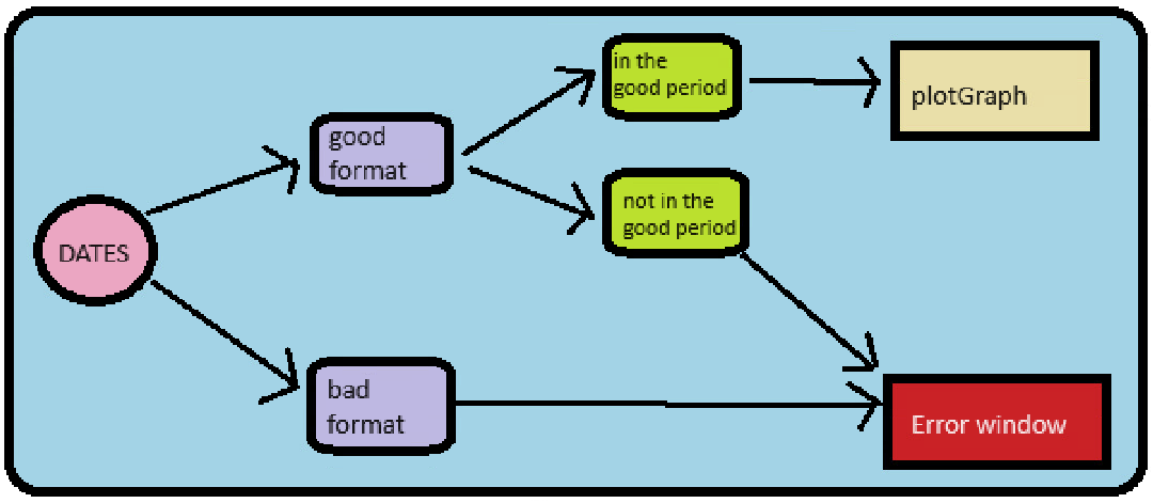

Figure 4 – `ChoixDateBouton` organisation

In this class, the program checks if the dates entered by the user are in the good format and in the observation period. If not, it opens an errow window.

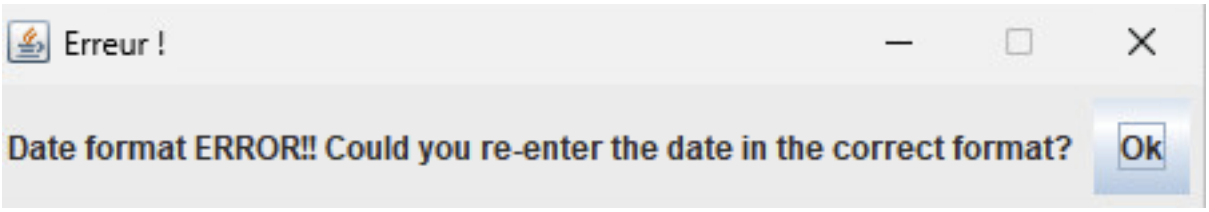

Figure 5 – Example of an error window when the dates format are not good

2.3 The class `PlotGraph`

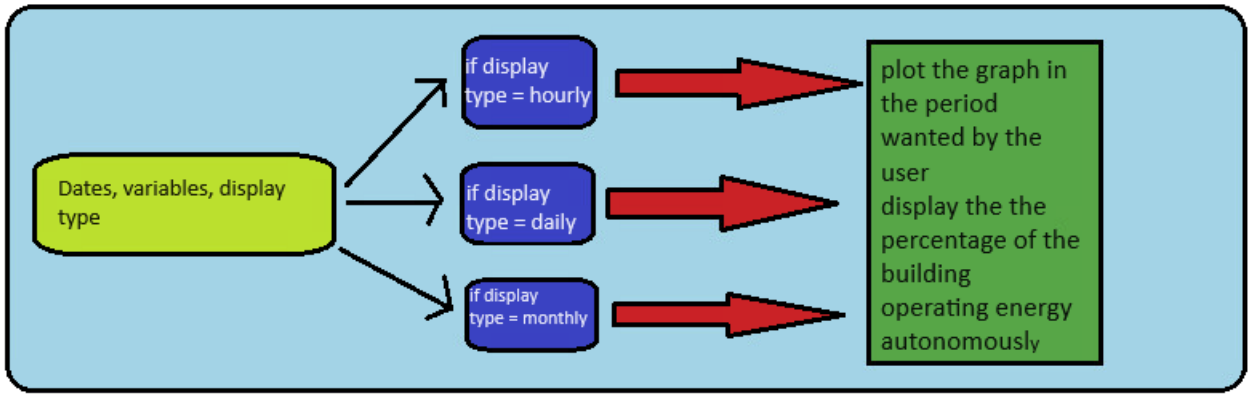

Figure 6 – `PlotGraph` organisation

Here, the program just display the graph of the selected variables by the user, how he wanted it (hourly, daily or monthly) and during the wanted period. To do the daily display (resp. monthly display), the program has to associate each day (resp. month) with the average values during that day (resp. month).

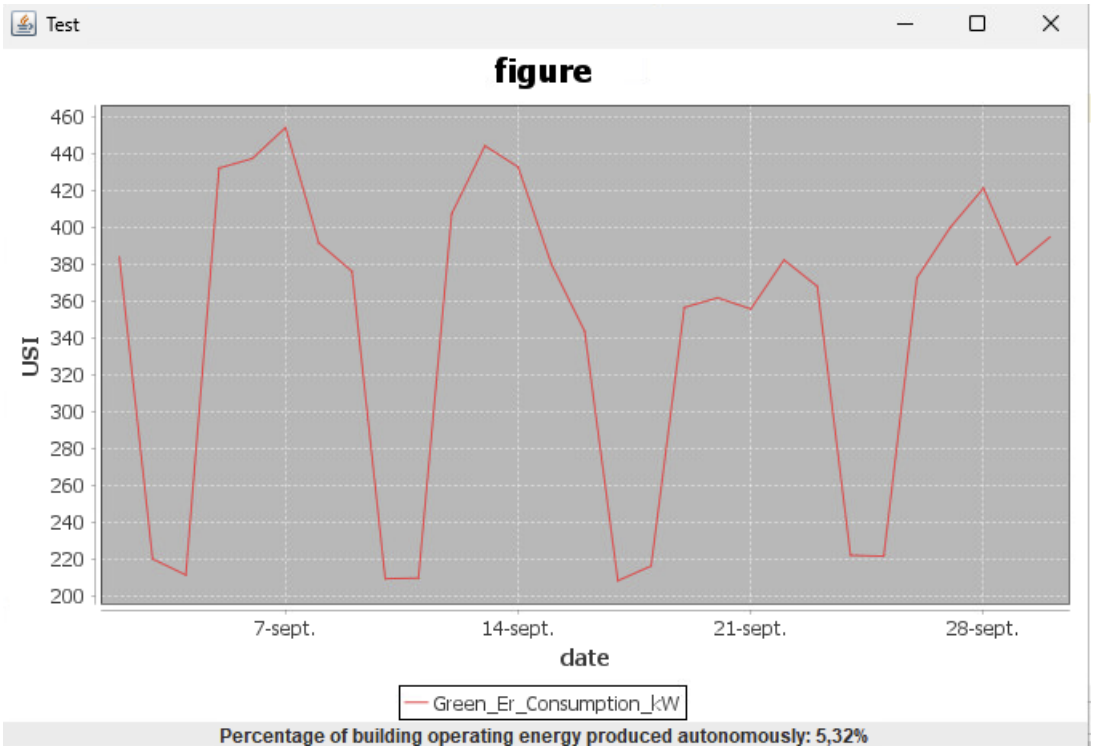

Figure 7 – Graph of `Gree_Er_Consumption_kW` during September 2022, display daily

A Annexes : Code

A.1 Main

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package pack;
import javax.swing.JFrame;
import java.awt.Graphics;
import java.io.IOException;
import javax.swing.SwingUtilities;
/**
 *
 * @author poteaum
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {
        // TODO code application logic here
        JFrame f = new FenPrincipale();
        //f.setVisible(true);
    }

}
```

A.2 DataContainer

```java
package pack;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.TreeMap;


public class DataContainer {

    // liste
    ArrayList<String> timeStrings;
    ArrayList<String> orderedVariableNames;
    TreeMap<String, ArrayList<Double>> data;
    TreeMap<String, ArrayList<String>> dataTimesString;
    int numberOfSamples = 0;

    /**
     *
     * @param csvFileName fichier de données à lire
     * @throws IOException
     */
    public DataContainer(String csvFileName) throws IOException {
        orderedVariableNames = new ArrayList<String>();
        data = new TreeMap<String, ArrayList<Double>>();
        dataTimesString = new TreeMap<String, ArrayList<String>>();
        int numberOfVariables = 0;

        timeStrings = new ArrayList<String>();
        ArrayList<String> orderedVariableNamesInFile = new ArrayList<>();
        int numberOfVariablesInFile = 0;
        BufferedReader bufferedReader = new BufferedReader(new FileReader(csvFileName));

        String line;
        line = bufferedReader.readLine();
        String[] tokens = line.split(",");

        for (int i = 1; i < tokens.length; i++) {
            orderedVariableNames.add(tokens[i]);
            orderedVariableNamesInFile.add(tokens[i]);
            data.put(tokens[i], new ArrayList<>());
            dataTimesString.put(tokens[i], new ArrayList<>());
            numberOfVariables++;
            numberOfVariablesInFile++;
        }
        int count = 0;
        while ((line = bufferedReader.readLine()) != null) {
            String[] values = line.split(",");
            if (values.length != 1) {
                for (int i = 0; i < numberOfVariablesInFile + 1; i++) {
                    if (i == 0) {
                        timeStrings.add(values[i]);
                        dataTimesString.get(orderedVariableNamesInFile.get(i)).add(values[i]);
                    } else {
                        data.get(orderedVariableNamesInFile.get(i - 1)).add(Double.parseDouble(values[i]));
                    }
                }
                count++;
            }
        }
        bufferedReader.close();

        numberOfSamples = timeStrings.size();
    }

    /**
     * retourne le nombre de samples
     *
     * @return number of samples
     */
    public int getNumberOfSamples() {
        return numberOfSamples;
    }


    /**
     * Retourne le nombre de variables
     * @return nombre de variables
     */
    public int getNumberOfVariables() {
        return data.size();
    }
```

```java
    /**
     * Récupère le temps d'observation sous forme de tableau de chaines de caractères
     * @return temps d'observation
     */
    public String[] getTimeStrings() {
        return timeStrings.toArray(new String[numberOfSamples]);
    }


    /**
     * récupère les dates sous forme de date
     * @return les dates
     * @throws ParseException
     */
    public Date[] getDates() throws ParseException {
        String val = "";
        Date[] dates = new Date[numberOfSamples];
        DateFormat format = new SimpleDateFormat("yyyy-MM-d HH:mm:ss");
        for (int i = 0; i < numberOfSamples; i++) {
            val = timeStrings.get(i);

            dates[i] = format.parse(val);
        }
        return dates;
    }



    /**
     * Donne les variables disponibles
     * @return variables disponibles
     */
    public String[] getAvailableVariables() {
        return orderedVariableNames.toArray(new String[getNumberOfVariables()]);
    }


    /**
     * Récupère les données associées à une variable
     * @param variableName
     * @return les données associéesn à variableName
     */
    public Double[] getData(String variableName) {
        return data.get(variableName).toArray(new Double[data.get(variableName).size()]);
    }


    /**
     * ajoute des données associées à une nouvelle variable dans data si ces données sont un tableau de double
     * @param variableName
     * @param values
     */
    public void addData(String variableName, double[] values) {
        if (values.length != getNumberOfSamples()) {
            throw new RuntimeException(variableName + " has " + values.length + " samples instead of " + getNumberOfSamples());
        }
        if (data.containsKey(variableName)) {
            throw new RuntimeException(variableName + " already exists");
        }
        orderedVariableNames.add(variableName);
        ArrayList<Double> newValues = new ArrayList<>();
        for (double value : values) {
            newValues.add(value);
        }
        data.put(variableName, newValues);
    }


    /**
     * ajoute des données associées à une nouvelle variable dans data si ces données sont un tableau de Double
     * @param variableName
     * @param values
     */
    public void addData(String variableName, Double[] values) {
        double[] primitiveValues = new double[values.length];
        for (int i = 0; i < values.length; i++) {
            primitiveValues[i] = values[i];
        }
        addData(variableName, primitiveValues);
    }

    @Override
    public String toString() {
        String string = getNumberOfVariables() + " variables: ";
        String firstRow = "[";
        String lastRow = "[";
        for (String variableName : getAvailableVariables()) {
```

```
            string += variableName + ", ";
            Double[] values = getData(variableName);
            firstRow += values[0] + ", ";
            lastRow += values[numberOfSamples - 1] + ", ";
        }
        string += "\nnumber of data: " + numberOfSamples + "\n";
        string += getTimeStrings()[0] + ": " + firstRow + "]\n...\n" + getTimeStrings()[numberOfSamples - 1] + ": " + lastRow + "]\n";
        return string;
    }

    /**
     * renvoie un tableau de dates contenant les jours d'observation sans redondance
     * @return les jours d'observation sans redondance
     * @throws ParseException
     */
    public Date[] getDailyDates() throws ParseException {
        ArrayList<String> dailyTimesString = new ArrayList<>();
        int j = 0;
        while ((j < this.numberOfSamples) && ((j + 24) < this.numberOfSamples)) {
            dailyTimesString.add(this.timeStrings.get(j));
            j+=24;
        }

        Date[] dates = new Date[dailyTimesString.size()];
        DateFormat format = new SimpleDateFormat("yyyy-MM-d HH:mm:ss");
        for (int i = 0; i < dailyTimesString.size(); i++) {
            dates[i] = format.parse(dailyTimesString.get(i));
        }
        return dates;
    }

    public static void main(String[] args) {
        try {
            DataContainer dataContainer = new DataContainer("GreenEr_data.csv");
            System.out.println(dataContainer);
        } catch (IOException e) {
            System.out.println(e);
        }
    }

}
```

A.3 FenPrincipale

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package pack;

import javax.swing.JFrame;
import java.awt.*;
import java.awt.event.WindowEvent;
import java.io.IOException;
import javax.swing.*;
import java.text.ParseException;
import java.util.ArrayList;
import javax.swing.JButton;
import javax.swing.JCheckBox;

/**
 *
 * @author poteaum
 */
public class FenPrincipale extends JFrame {


    /**
     * Affiche la fenetre principale
     * @throws IOException
     */
    public FenPrincipale() throws IOException {
        String minDate = "2022-08-31 00:00:00";
        String maxDate = "2023-09-01 23:00:00";
        setTitle("Affichage figures");
//        RecupDonnees recupDonnees = new RecupDonnees("GreenEr_data.csv");
        DataContainer recupDonnees = new DataContainer("GreenEr_data.csv");
        int nbVariales = recupDonnees.getNumberOfVariables();

        //setBounds(1, 1, 1000, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLayout(new BorderLayout());

        JPanel menu = new JPanel();
        JPanel choixConsumption = new JPanel();
        JPanel entreeDate = new JPanel();
        JPanel texteDateInit = new JPanel();
        JPanel texteDateFin = new JPanel();
        JPanel texteDate = new JPanel();
        /*JPanel centerPanel = new JPanel();
        JPanel eastPanel = new JPanel();
        JPanel southPanel = new JPanel();*/
        menu.setLayout(new BoxLayout(menu, BoxLayout.Y_AXIS));
        choixConsumption.setLayout(new FlowLayout());
        ArrayList<JCheckBox> listCheckbox = new ArrayList<JCheckBox>();
        for (String variableName : recupDonnees.getAvailableVariables()) {
            JCheckBox cb = new JCheckBox(variableName, false);
            choixConsumption.add(cb);
            listCheckbox.add(cb);
        }
        JCheckBox consMoinsProd = new JCheckBox("Green_Er_Consumption_kW - Green_Er_production_kW", false);
        choixConsumption.add(consMoinsProd);
        listCheckbox.add(consMoinsProd);



        for (JCheckBox cb : listCheckbox) {
            if (cb.isSelected()) {
                System.out.println(cb.getText());
            }
        }

        menu.add(choixConsumption);

        JPanel choixAffichage = new JPanel(new GridLayout(3,0));
        choixAffichage.setLayout(new FlowLayout());
        choixAffichage.add(new Label("Choose the display mode : "));
        JCheckBox choixHour = new JCheckBox("hourly", true);
        JCheckBox choixDay = new JCheckBox("daily", false);
        JCheckBox choixMonth = new JCheckBox("monthly", false);
        ButtonGroup checkBoxGroup = new ButtonGroup();
        checkBoxGroup.add(choixHour);
        checkBoxGroup.add(choixDay);
        checkBoxGroup.add(choixMonth);
        choixAffichage.add(choixHour);
        choixAffichage.add(choixDay);
        choixAffichage.add(choixMonth);
        ArrayList<JCheckBox> listAffichage = new ArrayList<JCheckBox>();
        listAffichage.add(choixDay);
        listAffichage.add(choixHour);
        listAffichage.add(choixMonth);
        menu.add(choixAffichage);



        texteDate.setLayout(new BoxLayout(texteDate, BoxLayout.Y_AXIS));
        entreeDate.setLayout(new FlowLayout());
        texteDateInit.setLayout(new FlowLayout());
        texteDateFin.setLayout(new FlowLayout());
        entreeDate.add(new Label("Veuillez ajouter les dates désirées"), BorderLayout.CENTER);
        texteDateInit.add(new Label("Initial date (format : yyyy-MM-dd HH:mm:ss)"));
        JTextField dateMinEntree = new JTextField("yyyy-MM-dd HH:mm:ss");
        texteDateInit.add(dateMinEntree);
        texteDateFin.add(new Label("Final date (format : yyyy-MM-dd HH:mm:ss)"));
        JTextField dateMaxEntree = new JTextField("yyyy-MM-dd HH:mm:ss");
        texteDateFin.add(dateMaxEntree);

        texteDate.add(texteDateInit);
        texteDate.add(texteDateFin);
        entreeDate.add(texteDate);
```

```
        JButton plotButton = new JButton("plot");
        JFrame fen = new JFrame("Erreur !");
        plotButton.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
        fen.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        fen.setLayout(new FlowLayout());
        plotButton.addActionListener(new ChoixDateBouton(minDate, maxDate, dateMinEntree, dateMaxEntree, fen, plotButton, listCheckbox, recupDonnees, listAffichage));
        //plotButton.addActionListener(new ChoixDateBouton(minDate, maxDate, dateMaxEntree, fen, plotButton));
        entreeDate.add(plotButton);
        menu.add(entreeDate);
        //menu.add(plotButton);
        add(menu);
        //fen.setVisible(true);
        while (fen.isVisible()) {
            plotButton.setEnabled(false);
            System.out.println(true);
        }

        /*for (JCheckBox cb : listCheckbox) {
            if(cb.isSelected()){
                System.out.println(cb.getText());
            }
        }*/
        setVisible(true);
        pack();

    }

}
```

A.4 ChoixDateBouton

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package pack;
import java.awt.BorderLayout;
import java.awt.event.*;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import javax.swing.*;

/**
 *
 * @author poteaum
 */
public class ChoixDateBouton implements ActionListener {
    String minDate;
    String maxDate;
    JTextField date1;
    JTextField date2;
    JFrame fen;
    JButton boutonRef;
    ArrayList<JCheckBox> listCheckbox;
    DataContainer recupDonnees;
    ArrayList<JCheckBox> listAffichage;
    /**
     * Vérifie que les dates sont au bon format et entre la plage d'observation
     * @param minDate date minimale plage de donnée
     * @param maxDate date maximale plage de donnée
     * @param date1 date minimale entree par utlisateur
     * @param date2 date maximale entree par utlisateur
     * @param fen fenetre d'affichage des erreurs
     * @param bouton bouton plot de la fenetre principale
     * @param listCheckbox liste des variables, on peut ainsi récupérer les checkbox qui ont été cochées
     * @param recupDonnees l'ensemble des données contenues dans "GreenEr_data.csv"
     * @param listAffichage liste des modes d'affichage des courbes (hour, month, day) on peut ainsi récupérer la checkbox qui a été cochée
     */
    public ChoixDateBouton(String minDate, String maxDate,JTextField date1, JTextField date2, JFrame fen, JButton bouton, ArrayList<JCheckBox> listCheckbox, DataContainer recupDonnees
       //System.out.println(date1);
       System.out.println(minDate);

       this.minDate=minDate;
       this.maxDate=maxDate;
       this.date1=date1;
       this.date2=date2;
       this.fen = fen;
       this.boutonRef = bouton;
       this.listCheckbox = listCheckbox;
       this.recupDonnees = recupDonnees;
       this.listAffichage = listAffichage;

    }

    @Override
    public void actionPerformed(ActionEvent e) {

        //panLabel.setLayout(new BoxLayout(panLabel, BoxLayout.Y_AXIS));
        fen.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
        fen.setResizable(false);


        JButton okButton = new JButton("Ok");
        okButton.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
        okButton.addActionListener(new OkErreurBouton(fen));
        //JFrame fen = new JFrame();


        try { // Si le format est bon
            DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            Date minD = format.parse(this.minDate);
            Date maxD = format.parse(this.maxDate);
            Date dateText1 = format.parse(this.date1.getText());
            Date dateText2 = format.parse(this.date2.getText());
            System.out.println(minD);
            String messageErreur1 = "";
            String messageErreur2 = "";
            int countErreur1 = 0;
            int countErreur2 = 0;



                if (dateText1.after(minD) && dateText1.before(maxD)) {
                    System.out.println(minD);
                    ArrayList<String> nameCbSelected = new ArrayList<String>();
                    String nameDisplayType = "";
                   for (JCheckBox cb : listCheckbox) {
                        if(cb.isSelected()){
                            nameCbSelected.add(cb.getText());
                        }
                    }

                   for (JCheckBox dt : listAffichage) {
                       if (dt.isSelected()){
                           nameDisplayType = dt.getText();
                       }
                   }

                   PlotGraph plotGraph = new PlotGraph(nameCbSelected, recupDonnees, date1, date2, minD, maxD, nameDisplayType);
                } else{
                    countErreur1++;
                    messageErreur1 = messageErreur1 + " initial date ";

                }
                if (dateText2.after(minD) && dateText2.before(maxD)) {
                } else{
                    countErreur2++;
                    messageErreur2 = messageErreur2 + " final date ";

                }

                int countErreur = countErreur1 + countErreur2;

                if (countErreur != 0){
                    String messageErreur = "";

                    if (countErreur == 2){
```

```
                    messageErreur = messageErreur1+ " and " + messageErreur2 + " are ";
                } else {
                    if (countErreur1 == 1){
                        messageErreur = messageErreur1+ " is ";
                    }
                     if (countErreur2 == 1){
                        messageErreur = messageErreur2+ " is ";
                    }
                }


                fen.getContentPane().removeAll();
                JLabel label = new JLabel("Problem with the dates");
                JPanel panLabel = new JPanel();
                panLabel.setBorder(BorderFactory.createEmptyBorder(50, 50, 50, 50));

                label.setText("NO!! the" + messageErreur + "not included between the two dates");

                fen.add(label);
                fen.add(okButton);

                fen.pack();
                fen.setVisible(true);
                boutonRef.setEnabled(false);
            }


        } catch (ParseException ex) {
            fen.getContentPane().removeAll();

            JLabel label = new JLabel("Problem with the dates");

            label.setText(" Date format ERROR!! Could you re-enter the date in the correct format? ");

            fen.add(label);
            fen.add(okButton);
            fen.pack();
            fen.setVisible(true);
            boutonRef.setEnabled(false);
        }
        fen.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosed(WindowEvent e) {
                //fen.dispose();
                boutonRef.setEnabled(true);


            }


        });






    }
}
```

A.5 OkErreurBouton

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package pack;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JTextField;

/**
 *
 * @author poteaum
 */
public class OkErreurBouton implements ActionListener{
    JFrame fen;
    public OkErreurBouton(JFrame fen){
       this.fen = fen;


    }

    @Override
    public void actionPerformed(ActionEvent e) {
        this.fen.dispose();
    }

}
```

A.6 PlotGraph

```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package pack;

import java.text.DecimalFormat;
import javax.swing.JLabel;
import java.awt.BorderLayout;
import java.util.ArrayList;
import javax.swing.JTextField;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TreeMap;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.data.time.Day;
import org.jfree.data.time.Hour;
import org.jfree.data.time.Month;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;

/**
 *
 * @author poteaum
 */
public class PlotGraph {
    ArrayList<String> listVariables;
    DataContainer recupDonnees;
    JTextField dateMin;
    JTextField dateMax;
    Date minD;
    Date maxD;
    String nameDisplayType;



    /**
     * Affiche le graph sur la plage demandé par l'utlisateur
     * @param listVariables liste des variables qui ont été sélectionnées par l'utilisateur
     * @param recupDonnees ensemble des données contenues dans "GreenEr_data.csv"
     * @param dateMin date minimale entree par utlisateur
     * @param dateMax date maximale entree par utlisateur
     * @param minDate  date minimale plage de donnée
     * @param maxDate   date maximale plage de donnée
     * @param nameDisplayType  variable qui dit si l'affichage se fera par heure, par jour ou par mois
     * @throws ParseException
     */
    public PlotGraph(ArrayList<String> listVariables, DataContainer recupDonnees, JTextField dateMin, JTextField dateMax, Date minDate, Date maxDate, String nameDisplayType) throws Pa
        this.listVariables = listVariables;
        this.recupDonnees = recupDonnees;
        this.dateMin = dateMin;
        this.dateMax = dateMax;
        this.minD = minD;
        this.maxD = maxD;
        this.nameDisplayType = nameDisplayType;


        DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        Date dateTextMin = format.parse(this.dateMin.getText());

        Date dateTextMax = format.parse(this.dateMax.getText());
        Date[] toutesDates = recupDonnees.getDates();
        /*for (Date date : toutesDates){
            System.out.println(date);
        }*/
        TimeSeriesCollection timeSeriesCollection = new TimeSeriesCollection();
        Double[] values = new Double[recupDonnees.numberOfSamples];
        double totalCons = 0.0;
        double totalProd = 0.0;
        Double[] valCons = this.recupDonnees.getData("Green_Er_Consumption_kW");
        Double[] valProd = this.recupDonnees.getData("Green_Er_production_kW");
        switch(nameDisplayType){
            case "daily":
                TreeMap<Day, ArrayList<Double>> dailyValues = new TreeMap<>();
                for(String var : listVariables) {
                    if ("Green_Er_Consumption_kW - Green_Er_production_kW".equals(var)) {

                        for(int i=0; i<recupDonnees.numberOfSamples; i++) {
                            values[i] = valCons[i] - valProd[i];
                        }
                    } else {
                        values = this.recupDonnees.getData(var);
                    }



                    for(int i=0; i<recupDonnees.numberOfSamples; i++){
                        if (toutesDates[i].after(dateTextMin) && toutesDates[i].before(dateTextMax)){
                            totalCons += valCons[i];
                            totalProd += valProd[i];
                            Day day = new Day(toutesDates[i]);
                            if (!dailyValues.containsKey(day)) {
                                dailyValues.put(day, new ArrayList<>());
                            }
                            dailyValues.get(day).add(values[i]);
                        }
                    }
                    TimeSeries timeSeries = new TimeSeries(var);
                    for (Day day : dailyValues.keySet())  {
                        ArrayList<Double> dailyData = dailyValues.get(day);
                        double average = dailyData.stream().mapToDouble(Double::doubleValue).average().orElse(-1);
                        timeSeries.add(day,average);
                    }
                    timeSeriesCollection.addSeries(timeSeries);
                }


                break;
            case "monthly" :
                TreeMap<Month, ArrayList<Double>> monthlyValues = new TreeMap<>();
                for(String var : listVariables) {
                    if ("Green_Er_Consumption_kW - Green_Er_production_kW".equals(var)) {

                        for(int i=0; i<recupDonnees.numberOfSamples; i++) {
                            values[i] = valCons[i] - valProd[i];
```

```
                    }
                } else {
                    values = this.recupDonnees.getData(var);
                }


                for(int i=0; i<recupDonnees.numberOfSamples; i++){
                    if (toutesDates[i].after(dateTextMin) && toutesDates[i].before(dateTextMax)){
                        totalCons += valCons[i];
                        totalProd += valProd[i];
                        Month month = new Month(toutesDates[i]);
                        if (!monthlyValues.containsKey(month)) {
                            monthlyValues.put(month, new ArrayList<>());
                        }
                        monthlyValues.get(month).add(values[i]);
                    }
                }
                TimeSeries timeSeries = new TimeSeries(var);
                for (Month month : monthlyValues.keySet())  {
                    ArrayList<Double> monthlyData = monthlyValues.get(month);
                    double average = monthlyData.stream().mapToDouble(Double::doubleValue).average().orElse(-1);
                    timeSeries.add(month,average);
                }
                timeSeriesCollection.addSeries(timeSeries);
            }
            break;
        default :
            for(String var : listVariables) {
                if ("Green_Er_Consumption_kW - Green_Er_production_kW".equals(var)) {

                    for(int i=0; i<recupDonnees.numberOfSamples; i++) {
                        values[i] = valCons[i] - valProd[i];
                    }
                } else {
                    values = this.recupDonnees.getData(var);
                }

            TimeSeries timeSeries = new TimeSeries(var);
            for(int i = 0; i < recupDonnees.numberOfSamples; i++) {
                if (toutesDates[i].after(dateTextMin) && toutesDates[i].before(dateTextMax)) {
                    totalCons += valCons[i];
                    totalProd += valProd[i];
                    timeSeries.add(new Hour(toutesDates[i]), values[i]);
                }
            }
                timeSeriesCollection.addSeries(timeSeries);
            }
            break;
    }

        double perAutoEnergy = (totalProd / totalCons) * 100;
        DecimalFormat df = new DecimalFormat("0.00");
        JPanel chartPanel = new ChartPanel(ChartFactory.createTimeSeriesChart("figure","date","USI",timeSeriesCollection,true, true, false));
        JFrame frame = new JFrame("Test");
        frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        JLabel autonomyLabel = new JLabel("Percentage of building operating energy produced autonomously: " + df.format(perAutoEnergy) + "%");
        autonomyLabel.setHorizontalAlignment(JLabel.CENTER);
        JPanel contentPane = new JPanel(new BorderLayout());

        contentPane.add(chartPanel, BorderLayout.CENTER);
        contentPane.add(autonomyLabel, BorderLayout.SOUTH);
        frame.setContentPane(contentPane);
        frame.pack();
        frame.setVisible(true);
    }


}
```

List of Figures