

**UNIVERSITÉ GRENOBLE ALPES**  
**INSTITUTE POLYTECHNIQUE DE GRENOBLE - Ense3**  
4EUS3ITO - IT Tools and Optimization

ALVES DOS SANTOS ASFORA Arthur

**BE - Problem: Green-Er consumption application**

Prof: RAMDANE Brahim  
Prof: FREISEM Léonard

**Grenoble - France**  
**Janvier 2025**

**Index**

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Application screens</b>	<b>3</b>
2.1	Green-Er Data . . . . .	4
2.2	classRoom_4A020 . . . . .	6
<b>3</b>	<b>Explanation of some methods</b>	<b>8</b>
<b>4</b>	<b>References</b>	<b>10</b>

# 1 Introduction

The Green-Er Energy Consumption Project that was developed focuses on developing a software tool in Java for managing energy data related to the Green-Er building. This building is equipped with advanced sensors to monitor energy consumption and production, particularly from photovoltaic panels installed on the roof.

Additional sensors in classroom G-4-A020 track energy usage of laptops, lighting, and temperature. The data provided spans the period from September 1, 2022, to August 31, 2023, with a 1-hour time resolution. We have implemented a function to change the sampling time.

The aim of the project is to design an application that processes and visualizes this data through a graphical user interface (GUI). The GUI will enable users to:

- Generate plots for energy measurements over specific periods.
- Analyze energy production and consumption differences.
- Offer flexible viewing options, such as daily or monthly resolutions.
- Provide insights into autonomous building energy performance and correlations with classroom sensor data.

## 2 Application screens

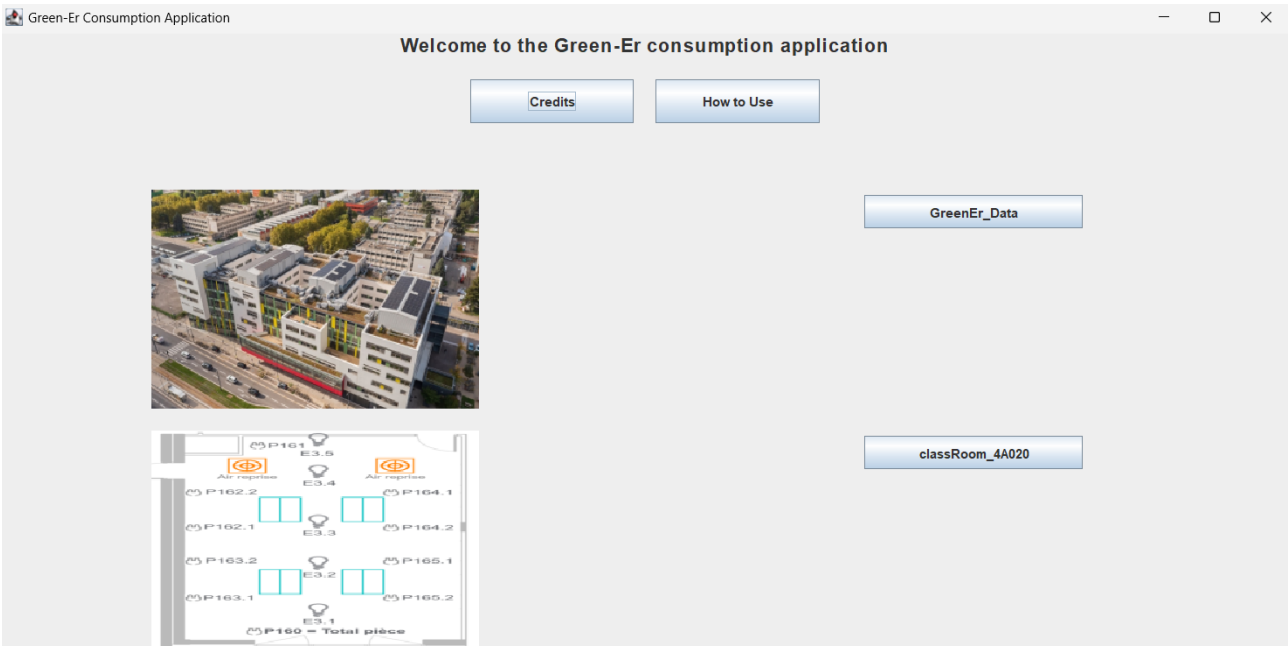


Figure 1: Welcome Screen

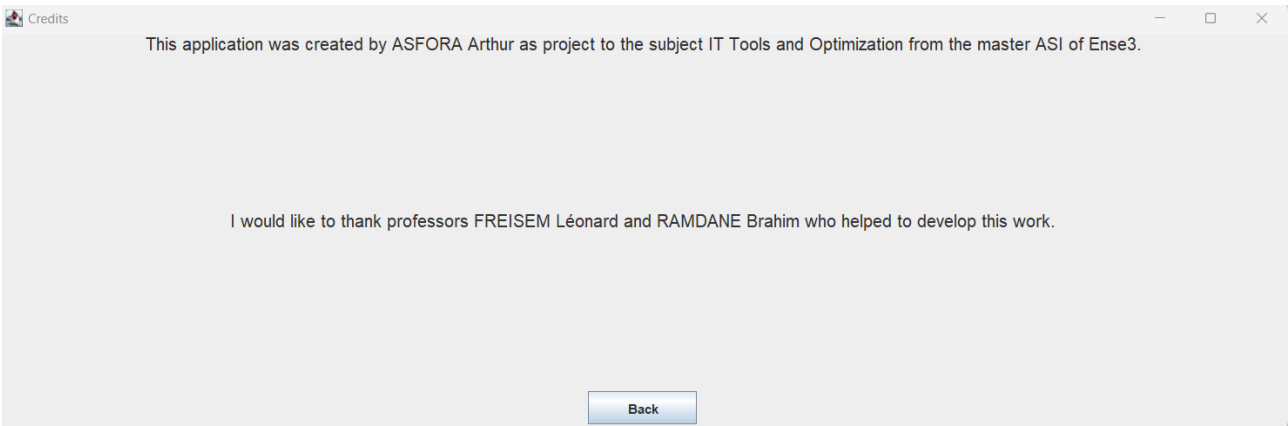


Figure 2: Credits Screen

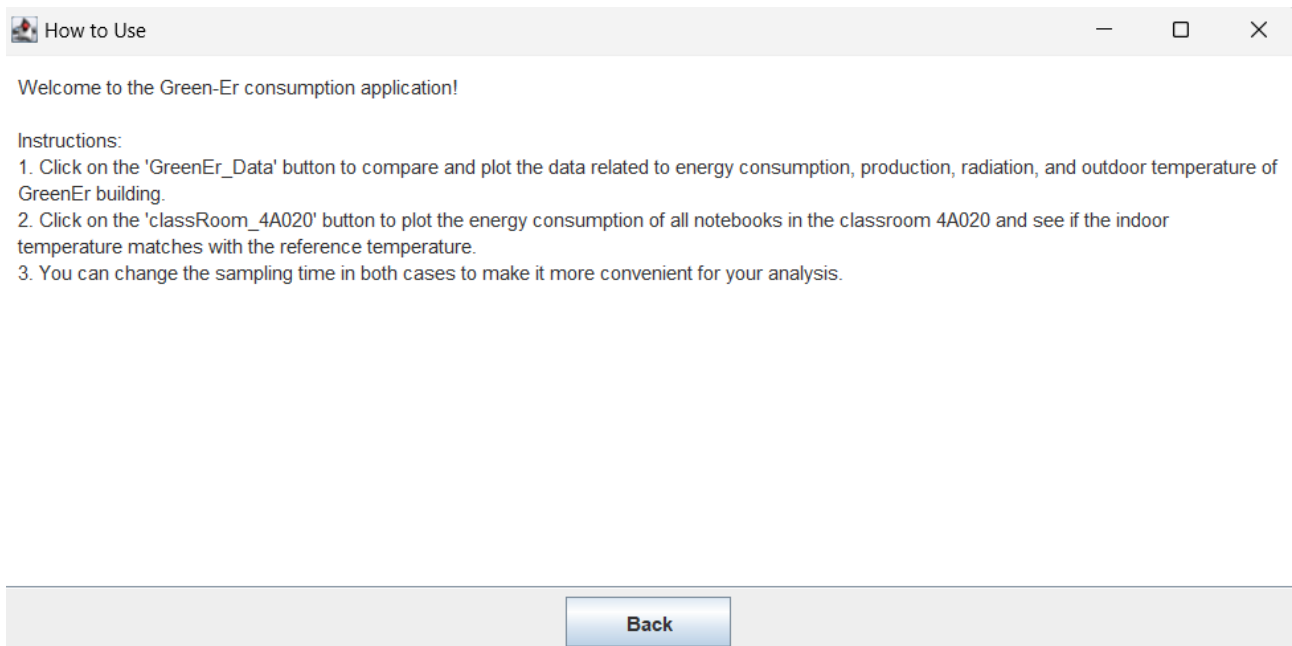


Figure 3: How to Use Screen

## 2.1 Green-Er Data

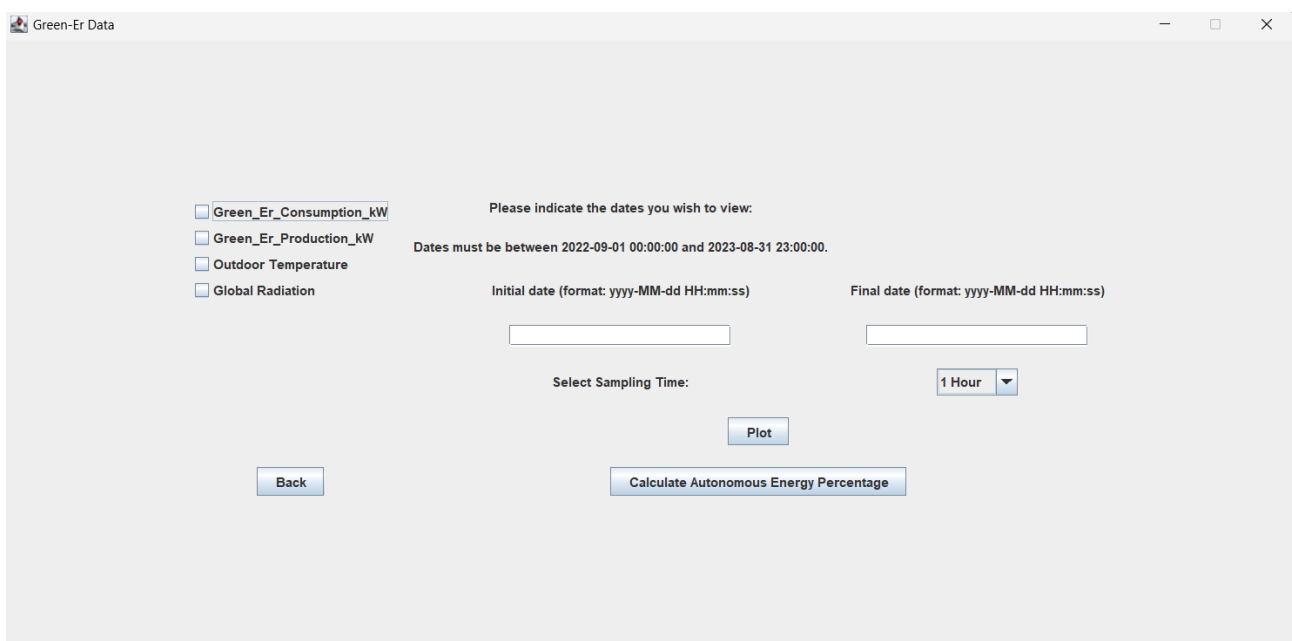


Figure 4: Green-Er Data Main Screen

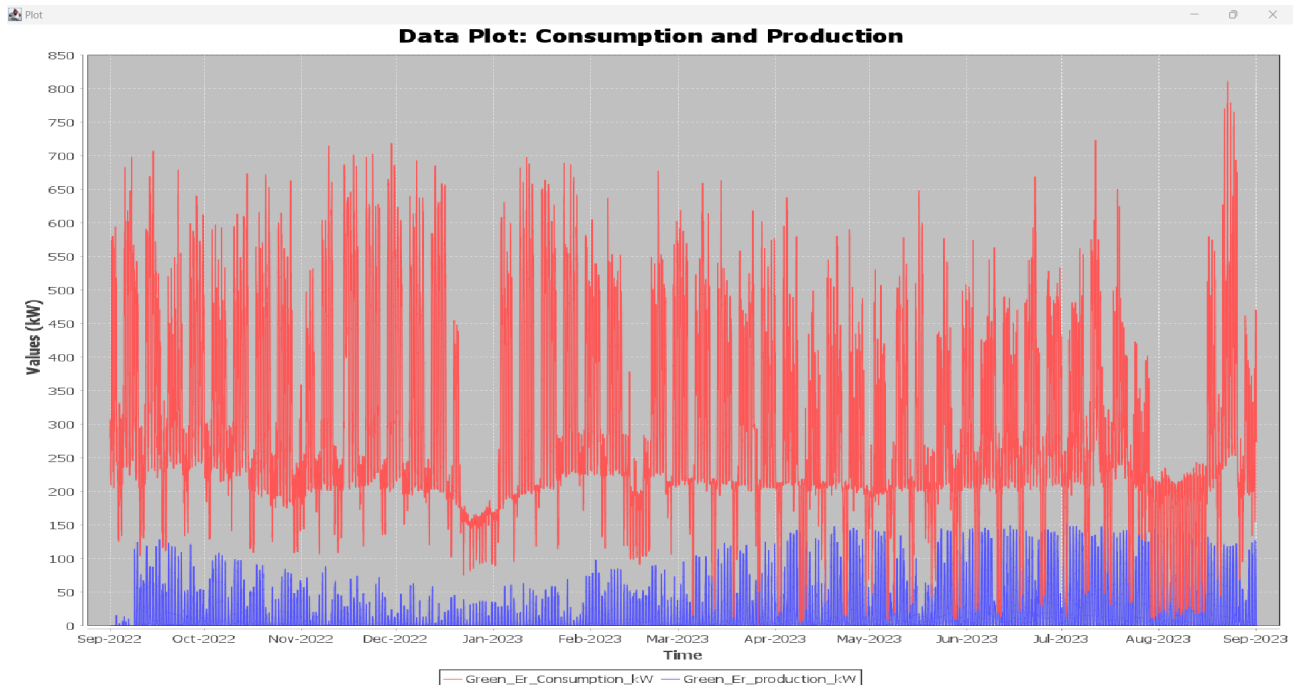


Figure 5: Plot Example

It is possible to change the sampling time. For the above graphic, we choose the default sampling time of 1 hour. Also, it is also possible to obtain the Autonomous Energy Percentage:

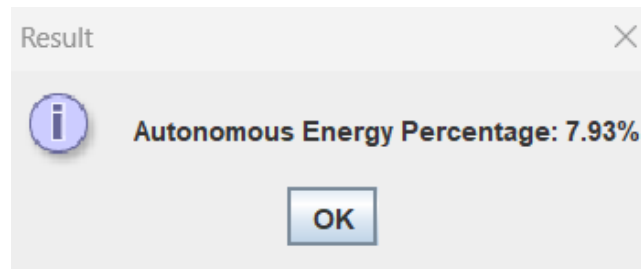


Figure 6: Autonomous Energy Percentage

## 2.2 classRoom\_4A020

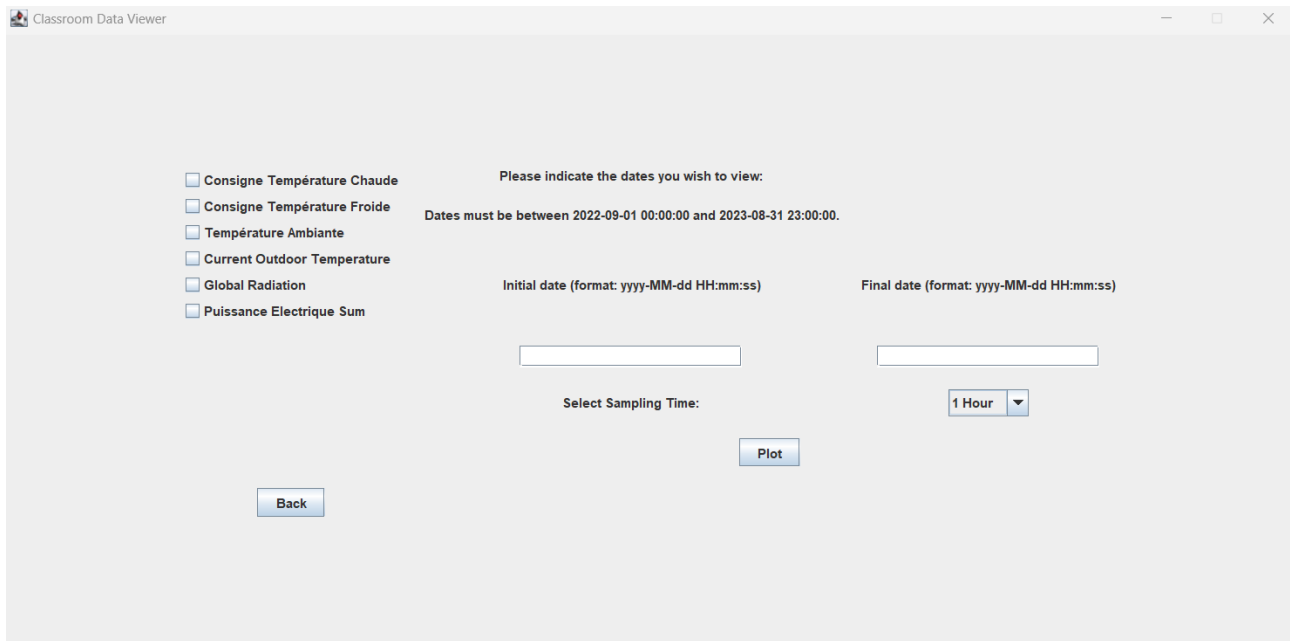


Figure 7: classRoom\_4A020 Main Screen

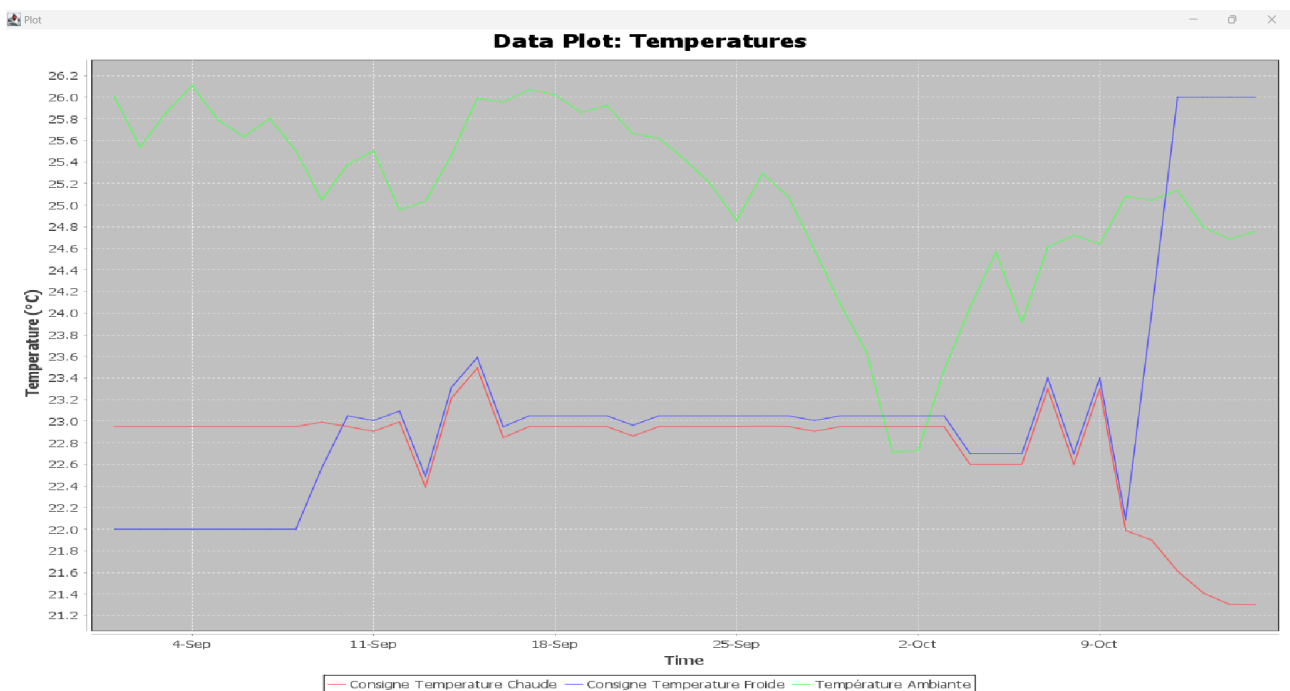


Figure 8: Plot Example

The figure above shows a plot example with a sampling time of 1 day between the dates 2022-09-01 00:00:00 and 2022-10-15 00:00:00. We've chosen three variables (hot and cold references and room temperature). It is also possible to plot four or two of them.

Correlating the total electric consumption of room laptops with the Global Radiation of photovoltaic cells (PV cells):

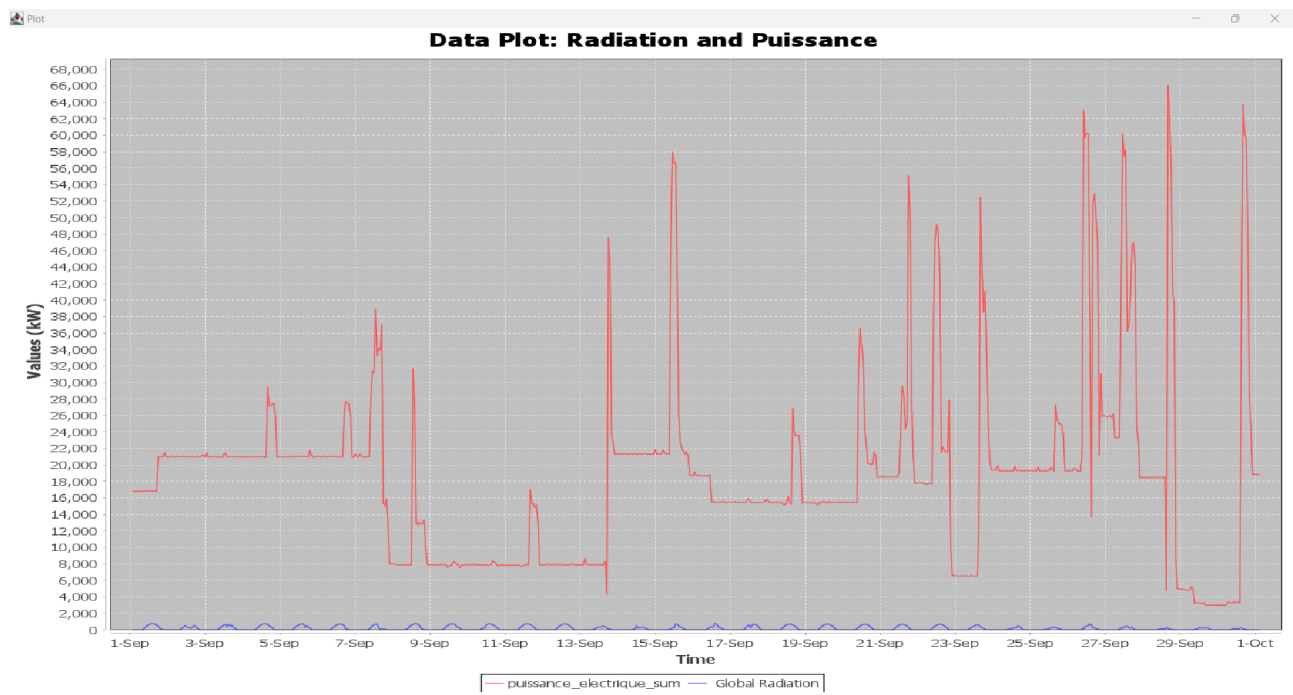


Figure 9: Plot Example 2



### 3 Explanation of some methods

For `DataContainer.java`, there are some methods that are worth explaining in detail how they were implemented.

#### **`public void computePuissanceElectriqueSum()`**

This method computes the total electrical power of laptops consumed by summing all variables with names beginning with `"puissance_electrique"`.

- It iterates through the dataset for each timestamp, identifies relevant variables, and calculates their sum.
- A new variable, `"puissance_electrique_sum"`, is created in the dataset to store these calculated values.
- This feature is particularly useful for summarizing energy usage across multiple devices.
- The method ensures modularity by dynamically detecting variables with the specified prefix, making it adaptable to changes in dataset structure.
- By storing the computed results directly in the dataset, it enables seamless integration with other data analysis operations.

#### **`public DataContainer resampleData(String samplingInterval)`**

The `resampleData` method adjusts the data's granularity by grouping and averaging values based on the specified `samplingInterval`, such as `"1 Hour"`, `"1 Day"`, or `"1 Month"`.

- The method uses a concept called **buckets**, where each bucket represents a time interval (e.g., a day).
- Data points falling within the same bucket are aggregated, and their averages are computed for each variable.
- A `TreeMap` is used to store the aggregated data, ensuring efficient organization and access.
- The method dynamically determines bucket boundaries based on the sampling interval, leveraging `SimpleDateFormat` for time formatting.
- When transitioning between buckets, the current bucket's average values are computed and stored before moving to the next bucket.

#### **`public DataContainer filterByDateRange(String start, String end)`**

The `filterByDateRange` method selects data points within a specific date range, defined by the `start` and `end` parameters.

- Dates are parsed using `SimpleDateFormat`, which ensures compatibility with the input date format.
- Each timestamp is compared against the start and end boundaries, retaining only the data points within the range.
- A new `DataContainer` is created to store the filtered data, ensuring that the original dataset remains unmodified.

- All variables in the dataset are filtered simultaneously, maintaining consistency across time-related analyses.
- This method is crucial for focusing on specific periods, such as analyzing energy trends during peak hours or specific days.

## 4 References

1. Tutorial-Java-Course from Chamilo.
2. GeeksforGeeks: Java JFrame.