21.04.2023

BBM104 ASSIGNMENT 2 Report

ILHAN ARAS
B2210356023

HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

Index

The Problem Definition, System Explanation and Benefits of System

The aim of this project is to create a system that can efficiently handle smart home devices, such as Smart Lamps, Smart Plugs, and Smart Cameras. The system should enable users to control these devices based on time-based commands and arrange them in ascending order based on their switch times.

To achieve this goal, it is crucial to ensure that the system can handle exceptional situations, such as when two or more devices have the same switch time. In such cases, the original order of the devices should be maintained while sorting. Furthermore, devices without any switch time should be considered greater than all other devices.

The system should also allow users to change the device status, which will delete the switch time information. To ensure that the system is efficient and maintainable, it is important to design it using sound Object-Oriented Programming (OOP) principles, such as Inheritance, Polymorphism, Abstraction, and Encapsulation. The design should also consider any trade-offs.

In software development process, it is crucial to prioritize modularity, which involves breaking down a system into smaller, independent components. This makes it easier to isolate and test specific functions, identify and address issues, and add new features without disrupting the entire system. Utility classes can also enhance flexibility by providing reusable functions across various classes or modules. By prioritizing these principles, we can ensure that the system can adapt to future growth and changes without sacrificing stability or performance.

Some major problems of system are that manage devices with timeline, manage the database which contains important variables and find and classify the errors. The managing time and database management problems solved by using arraylist, hashmap and localdatetime module. Despite these solutions, there is a small error in the system. When 2 devices setted same time value and after the nop command, there is a small ordering bug. 2 devices which were assigned same time value are sorted reverse. however, despite this little most of the system work correctly. The benefits of this system are that it is flexible and changeable.

OOP Benefit and Four Pillars of OOP

OOP in Java has two main advantages: code reusability and modularity. With inheritance and polymorphism, developers can refine and improve code over time while preserving reusable code. Code modularity makes it easy to interchange and remove objects to meet specific user needs. OOP also leads to easier maintenance by minimizing the work required when making changes to a base class, reducing the occurrence of bugs and errors. Lastly, OOP breaks down complex concepts into real-life entities, improving communication between developers and users.

Object-oriented programming (OOP) is centered around encapsulation, inheritance, polymorphism, and abstraction. Encapsulation refers to the practice of consolidating data and methods in a single location. Inheritance facilitates the creation of new classes based on existing ones, eliminating the need for starting from scratch. Polymorphism permits objects to act differently based on varying circumstances. Abstraction streamlines code and minimizes repetition, enhancing its readability and ease of modification in the future.

UML Diagrams And Design Explanation

(To view a more comprehensive image, please click on or navigate to the final page.)

Accessories Class

The instance variables of the Accessories class consist of a String "name", a LocalDateTime "date", and a String "status". Getter and setter methods are publicly available for each instance variable to enable other parts of the program to retrieve and modify their values as needed.

Upon creating a new "Accessories" object, the "status" instance variable is initialized with the default value of "Off". The LocalDateTime "date" instance variable appears to be designed to store the date and time of an event or action related to the accessory object.

Overall, the code is structured in a well-organized manner and serves as a solid foundation for further Java programming.

SmartLamp Class

The Accessories class is subclassed by the SmartLamp class, with added properties and methods specific to a smart lamp. Different combinations of parameters can be passed when creating a new object, facilitated by three constructors. The default values of the kelvin and brightness properties are 4000 and 100, respectively, and can be modified using setter methods. Whether the lamp is in color mode or not is indicated by the isColorMode property, which is a boolean value.

A string representation of the SmartLamp object, including the name, status, kelvin value, brightness, and time to switch status, is provided by the toString() method. The formatter in the DataBase class is used to format the date. Within an accessories system, the SmartLamp class offers a flexible and customizable way to create and manage smart lamps.

SmartColorLamp Class

The SmartColorLamp class is a subclass of Accessories, with additional properties and methods specific to a smart color lamp device. It includes a protected String variable named colorcode, which is initialized to "0xFFFFFF". The class has four constructors to manage the properties of the lamp.

The first constructor takes a name parameter and calls the superclass constructor with that parameter. The second constructor takes name and status parameters and also calls the superclass constructor with the name parameter and sets the status variable to the given status. The third constructor takes name, status, kelvin, and brightness parameters and sets the respective variables in the superclass and this class. The fourth constructor takes name, status, colorcode, and brightness parameters, and sets the respective variables in the superclass and this class.

SmartPlug Class

The SmartPlug class is a subclass of Accessories, containing additional properties and methods that are specific to a smart plug device. It possesses several fields, including the voltage (volt), amperage (ampere), open and close times (LocalDateTime), total energy consumption (totalenergyconsumption), and plug status (plugstatus).

The class comprises three constructors, each of which takes a different set of parameters to initialize the object. The first constructor accepts only the name of the plug, while the second constructor also accepts a status parameter. If the status is "On", the open time is set to the global date in the database. The third constructor additionally accepts an ampere parameter.

The class has a method called calculateEnergyConsumption, which is responsible for calculating the energy consumption of the plug. This calculation is done by subtracting the close time from the open time, multiplying the result by the amperage, voltage, and time, and adding it to the total energy consumption field.

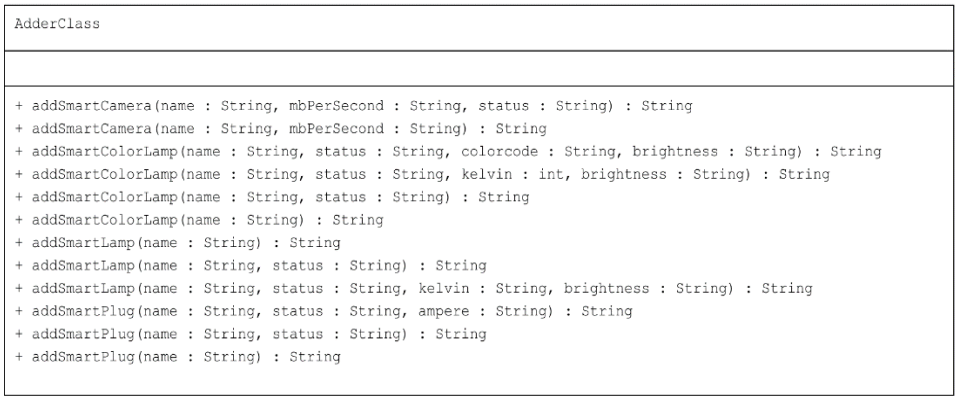
Several getter and setter methods are present in the class, which can be used to access and modify its fields. Finally, the toString method returns a string representation of the SmartPlug object, including its name, status, total energy consumption, and the time at which it is scheduled to switch its status.

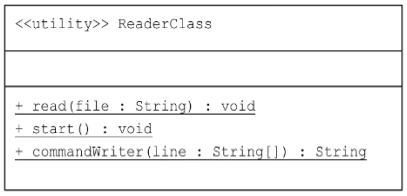
SmartCamera Class

The SmartCamera class, which is a subclass of Accessories, comprises of additional properties and methods that are specific to a smart camera device. The class has two constructors, one of which accepts a name and a mbPerSecond value as parameters, while the other takes a name, mbPerSecond, and a status string as parameters. If the status is "On", the openTime property is set to the global date and time value provided by the DataBase class. A method named calcMb() is included in the class, which is responsible for calculating the MB memory usage of the camera. This calculation is based on the difference in time between the openTime and closeTime properties, as well as the value of mbPerSecond. The outcome of this calculation is then added to the MBMemoryUsage property. The class also includes getter and setter methods for the mbPerSecond, openTime, and closeTime properties.

A string representation of the SmartCamera object, which comprises its name, status, MB memory usage, and the time at which it should switch its status based on the value of the date property provided by the DataBase class, is provided by the toString() method.

Adder Class and Reader Class

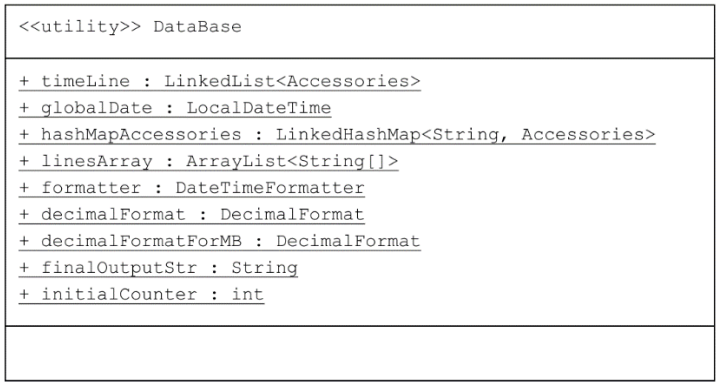




Several methods that add devices to the system are possessed by the Adder Class. This class has a "has a" relationship with other classes.

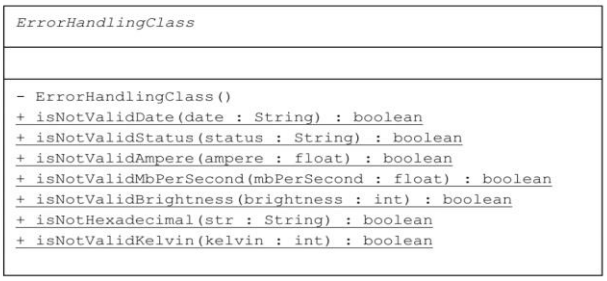
The Reader class reads an input file and adds data to an ArrayList. Additionally, it has a start method which initiates the program. In terms of performance and security while performing IO operations, data is immediately added to the ArrayList, and various operations are then carried out by using the ArrayList.

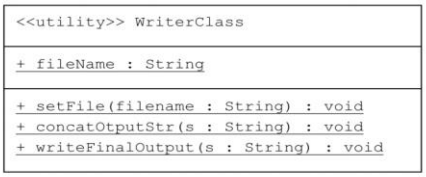
Database Class



Public static variables are contained within the database class. It is responsible for providing some common variables and functions that can be used by other classes.

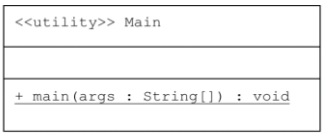
ErrorHandling Class and Writer Class





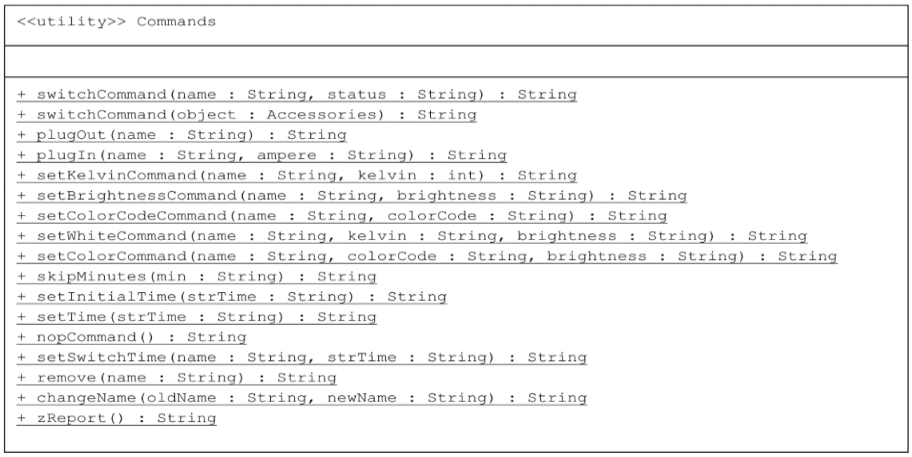
Some error finder methods are included in the ErrorHandling class. Additionally, there are some methods in the WriterClass which are utilized for writing data output into an output file.

Main Class

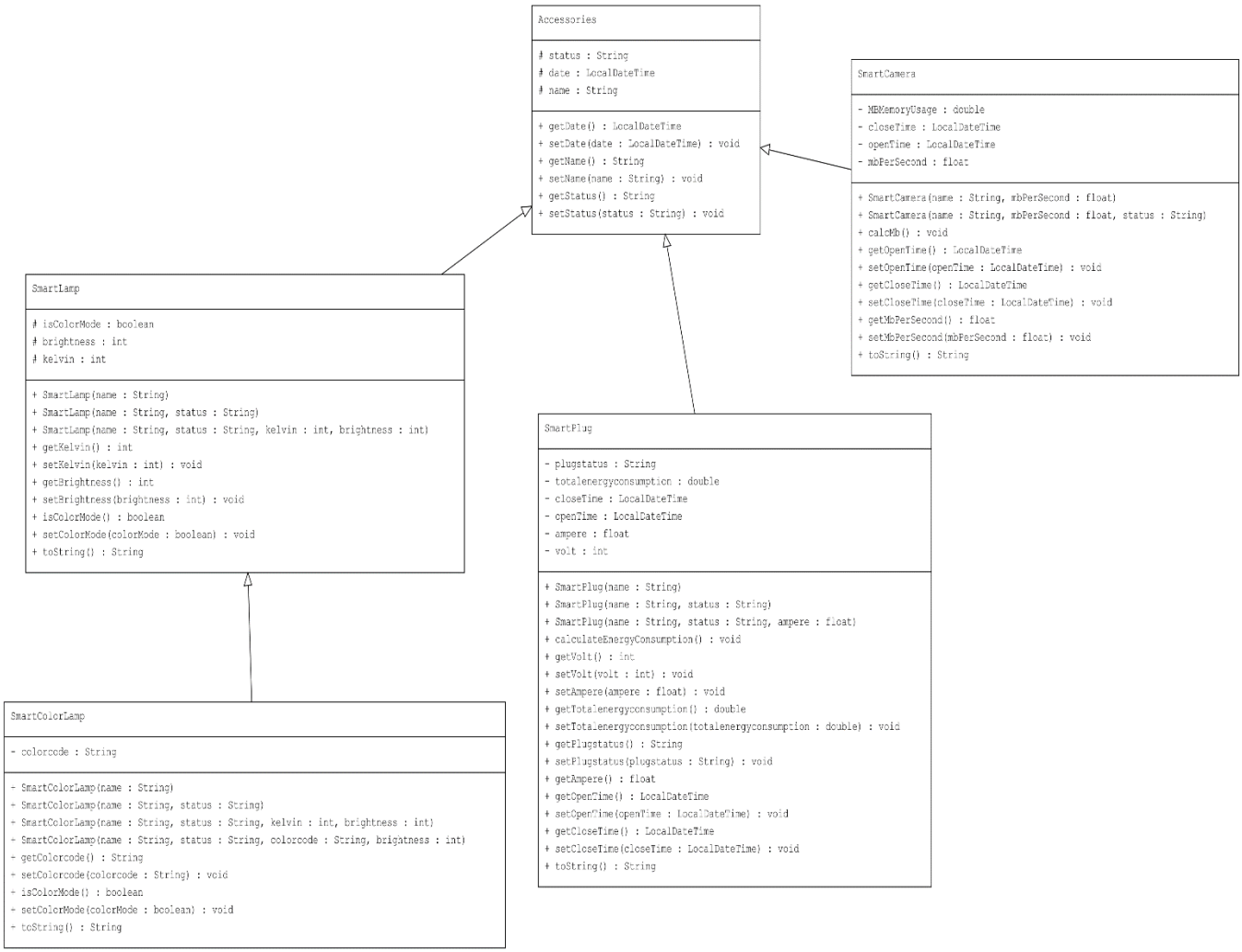


The main method is considered as a starting point for a program's execution, being acted upon as a gateway to its functionality and ensuring that the code fragments are properly executed.

Command Class



Several methods are possessed by CommandClass. These methods are deemed important to the program and are public static methods.


Accessories
status : String
date : LocalDateTime
name : String
+ getDate() : LocalDateTime
+ setDate(date : LocalDateTime) : void
+ getName() : String
+ setName(name : String) : void
+ getStatus() : String
+ setStatus(status : String) : void
SmartCamera
- MBKemoryUsage : double
- closeTime : LocalDateTime
- openTime : LocalDateTime
- mbPerSecond : float
+ SmartCamera(name : String, mbPerSecond : float)
+ SmartCamera(name : String, mbPerSecond : float, status : String)
+ calcMb() : void
+ getOpenTime() : LocalDateTime
+ setOpenTime(openTime : LocalDateTime) : void
+ getCloseTime() : LocalDateTime
+ setCloseTime(closeTime : LocalDateTime) : void
+ getMbPerSecond() : float
+ setMbPerSecond(mbPerSecond : float) : void
+ toString() : String
SmartLamp
isColorMode : boolean
brightness : int
kelvin : int
+ SmartLamp(name : String)
+ SmartLamp(name : String, status : String)
+ SmartLamp(name : String, status : String, kelvin : int, brightness : int)
+ getKelvin() : int
+ setKelvin(kelvin : int) : void
+ getBrightness() : int
+ setBrightness(brightness : int) : void
+ isColorMode() : boolean
+ setColorMode(colorMode : boolean) : void
+ toString() : String
SmartPlug
- plugstatus : String
- totalenergyconsumption : double
- closeTime : LocalDateTime
- openTime : LocalDateTime
- ampere : float
- volt : int
+ SmartPlug(name : String)
+ SmartPlug(name : String, status : String)
+ SmartPlug(name : String, status : String, ampere : float)
+ calculateEnergyConsumption() : void
+ getVolt() : int
+ setVolt(volt : int) : void
+ setAmpere(ampere : float) : void
+ getTotalenergyconsumption() : double
+ setTotalenergyconsumption(totalenergyconsumption : double) : void
+ getPlugstatus() : String
+ setPlugstatus(plugstatus : String) : void
+ getAmpere() : float
+ getOpenTime() : LocalDateTime
+ setOpenTime(openTime : LocalDateTime) : void
+ getCloseTime() : LocalDateTime
+ setCloseTime(closeTime : LocalDateTime) : void
+ toString() : String
SmartColorLamp
- colorcode : String
- SmartColorLamp(name : String)
- SmartColorLamp(name : String, status : String)
- SmartColorLamp(name : String, status : String, kelvin : int, brightness : int)
- SmartColorLamp(name : String, status : String, colorcode : String, brigatness : int)
- getColorcode() : String
- setColorcode(colorcode : String) : void
- isColorMode() : boolean
- setColorMode(colorMode : boolean) : void
- toString() : String