Hacettepe University

Computer Engineering Department

BBM102 PROJECT: SMART HOME SYSTEM - 2023 SPRING

Programming Assignment 2

April 20, 2023

Student name:
İlbey
GÜLMEZ

Student Number:
b2210765024

1 Problem Definition

In this project, it has been assigned to implement an OOP design based on a Smart Home System. To do this, four pillars of OOP will be used in the project.

2 Solution Implementation

Program consists of three steps mainly:

1) Creating the classes and methods

First, Time class is implemented, and approached as a static time flow rather than a dynamic algorithm. Then, the Accessory class representing the mutual attributes and methods of all Smart devices is implemented, it is planned to inherit all device types from the Accessory class.

After that, Smart Plug device is implemented and by getting the difference between the "On" and "Off" times and with the help of the formula, total energy consumption will be calculated. In Smart Camera, to calculate the megabyte value, the same logic will be used.

Later, in the second part, lamps are implemented. First, smart lamp is implemented and next smart color lamp inheriting the smart lamp is implemented. Finally, Commands class is implemented.

2) Implementing commands

For this, first the given text file is read and reached line by line, then by splitting it with respect to "\t," using switch – case and list indexing, the commands are read and executed.

Most of them can be called from device classes with small format arrangements, but there were some functions such as Remove, Change Name, Z Report. These functions should be created in this class. And later on, and the inevitable, Array List from Accessory class is called for help.

Added accessories are held in an Array List created in Accessory class, so given commands are executed on this Array List.

Finally, all commands are executed, Array List has correct devices with correct attributes. In Z Report they are printed in a specific order. So, it must be sorted with a Comparator with a custom logic implemented.

3) Debug- last touch

Finally, it has been gone through the program for bugs, unused variables etc.

Program came to final form.

3. Problems Encountered and Solutions

After Time class was successfully implemented, in Accessory.java, Switch and Switch Necessary methods was very troubling, since they are used all around the program, every inch of arrangement on them should be thought and formatted on a very detailed way, it's been solved by dividing the status attribute to statusBool – statusString, so program had less trouble formatting.

After that, in smart plug class, calculating energy consumption was extremely complicated since it was difficult to determine where the device was plugged in and switched on at the same time, it surely took more than an if block. I solved it by

rearranging PlugIn – PlugOut methods and overriding Switch method accordingly by adding startTime and endTime flags.

Later, during the implementation of the Smart Camera class, Switch method has been overridden again to modify the method accordingly. But calculation was never true, then it became clear that this time, time was in minutes (it was in hours in Smart Plug). So, it was solved like that.

Then, after successfully implementing the SmartLamp class, it was a problem determining if the given String is a hexadecimal or not. With a detailed method containing multiple format and parameter checks, it's been done.

Finally, the big bite – Commands class was implemented, it was harder than any part of the program. After reading the input file and checking the necessary conditions, it was possible to reach the commands in the desired format. Since program read the commands correctly, it was about time to execute them.

In each line, the program checked the first element of the line, with a switch—case operator, and it determined which command is desired and the operator executed them accordingly, problem was some errors caused by formats (or others), their message was printed in the Commands class, that resulted with program printing the error message but still does the erroneous execution on the object. That was a tough one, but it's been solved by adding different if blocks to certain places in the code.

After that, program ran into an another problem, multiple error messages. It's been solved by adding a Boolean representing the priority of errors. Finally, the biggest function was at hand, ZReport.

In ZReport, all device classes' toString() methods are called and used in the switch—case that was implemented to determine the type of the device in the accessories array list. The overridden methods are also used in Remove function.

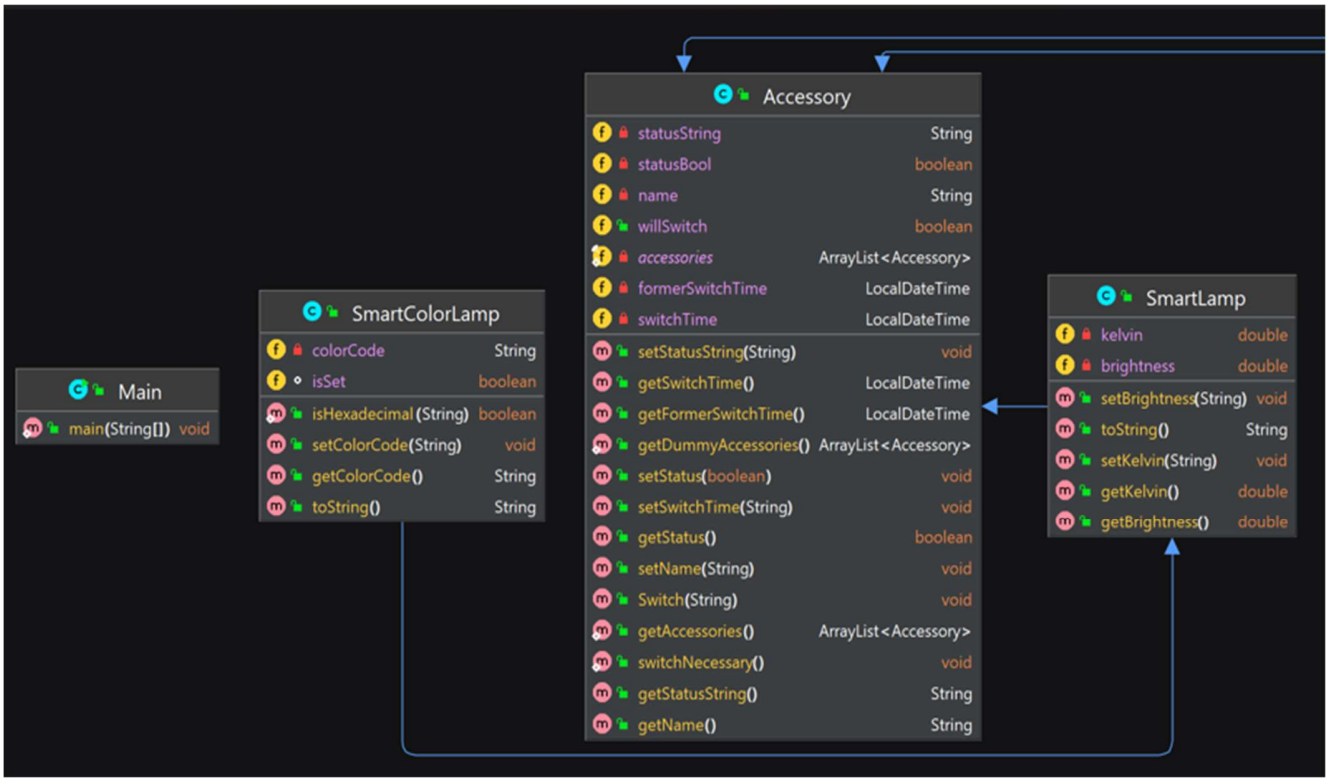
ZReport worked fine, except it was in the wrong order. After a lot of research (which felt like it should have been learnt from the classes) concerning comparators and even inner classes, a custom comparator was implemented. At first, it looked okay but later, the output seemed to put devices with null values at a constant order, but the desired output required that if the switch time of the device was later than the other one before switching (which will assign their switch time attributes to null), it should be put above the other. With an if block arrangement in the comparator class, it was solved.

4. Results, Analysis, Discussion

After completing the code, it was clear that this project was meant to teach inheritance (which is a property that enables classes to inherit properties from super classes) and polymorphism (which is a technique to use variables of several types on various times) above the other 2 pillars of OOP. Encapsulation (which is a technique of implementing classes so that the other developers or users use it comfortably) and Abstraction (which has the almost same use as encapsulation, but it is doing it by hiding the unnecessary details from the users) was not used as much as the others.

Consequently, by using OOP in this system, developers are saved from using repeated, long, complicated codes while implementing the system. If it weren't for OOP, the whole code would be complicated like Commands class.

5.UML Diagram Part 1



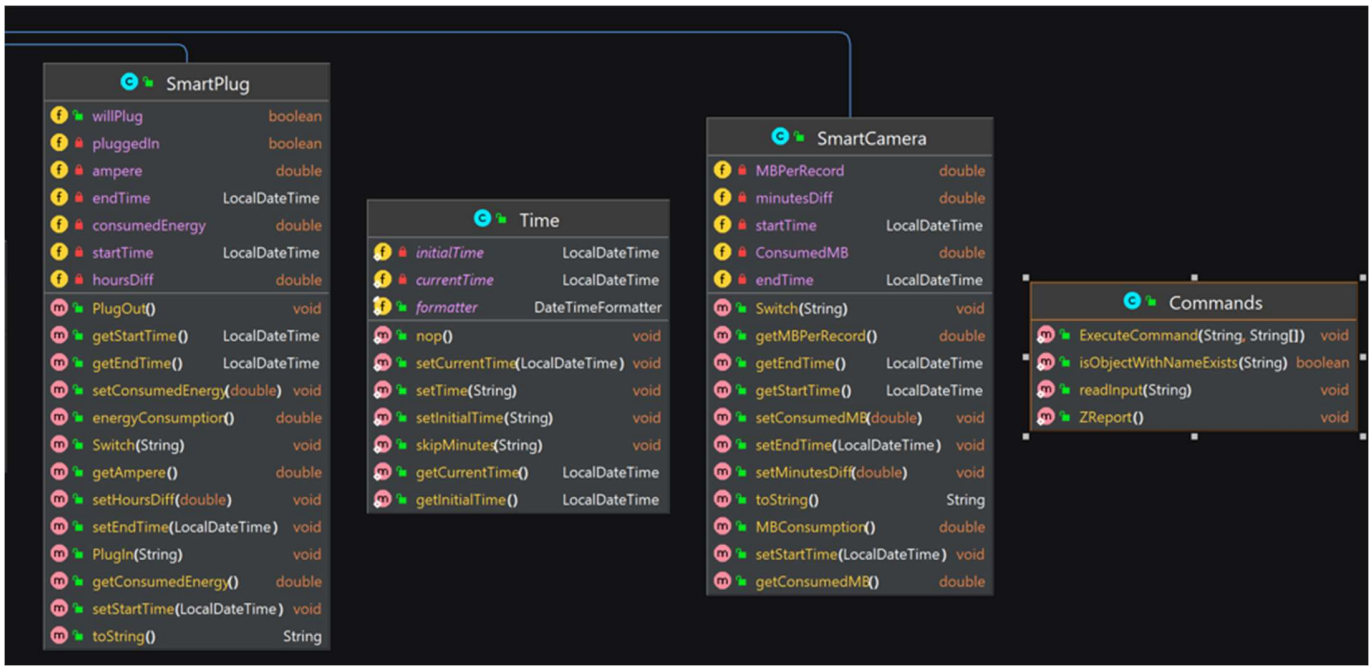
Explanation:

In this part of the UML diagram, SmartLamp and SmartColorLamp classes inherit from the Accessory class which contains the general attributes of the devices such as name, status, switchTime etc.

SmartLamp has kelvin and brightness attributes representing the lamp, and they have getter – setter methods for these attributes.

SmartColorLamp on the other hand, inherits from the SmartLamp class, so it also has the SmartLamp's attributes and colorCode attribute which represents a hexadecimal. isSet is a variable created to prevent multiple error messages. Getter – setters aside, SmartColorLamp has an isHexadecimal method that checks if a String is hexadecimal in the given range or not.

6.UML Diagram Part 2



Explanation:

In this final part of the UML diagram, SmartPlug and SmartCamera classes inherit from the Accessory class (arrows show it). SmartPlug has a plug attribute and relevant methods respectively, it also has a method to calculate the energy it consumes.

In SmartCamera, there is a similar dynamic as energy consumption, but this time, time is in minutes, and we calculate megabytes. Just like ampere, camera has a MBPerRecord attribute.

Then Time class is seen, this is a mostly static class that has variables used in methods all over the program including methods the Time class itself has.

Finally, the commands class is seen, it does not have any attributes respectively, only methods to execute the program.