HACETTEPE UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

BBM 104 ASSIGNMENT 2 REPORT

Ahmet İkbal Kayapınar

2210356043

22/04/2023

Index

Explanation of the Problem 2
Explanation of Solution Approach 2
Problems Encountered and Their Solutions 2
Benefits of the System 2
Benefits of OOP 2
Four Pillars of OOP 3
UML Diagram and Explanation of the UML Diagram 3

Explanation of the Problem

In this project, it is aimed to create a simple smart home system that can control 4 different types of smart devices with commands. If the program has any error in any command it receives, it will detect that error and continue to work without any problems. The information of each device can be entered while adding devices to the program or with subsequent commands. Although each type of device has its own commands, the program will also be able to create a general table showing the current status. The basis of the program is the time system it uses and the ability to manage the opening and closing of each device based on time.

Explanation of Solution Approach

In order for the program to manage devices efficiently, object-oriented programming must be used intensively. Therefore, in the program there is a basic device class that has the common features of all devices and classes that represent each device. The smart color lamp class, which is basically the same type with smart lamp class, is a subclass of the smart lamp class. Classes include common or device-specific methods as well as ensuring that data can be stored correctly. Classes apply the operations that devices do a lot in themselves to their own objects. The program will basically process the commands in the script it receives sequentially. Therefore, when each instruction comes, the program will distinguish it thanks to the conditional statement it has and call the appropriate methods for that instruction. The time system of the program that controls the opening and closing of the devices is basically controlled by 4 commands. The program has methods that make the necessary changes by controlling the system and devices at every moment of change.

At the same time, there are some rules of the program and many entries that can cause the program to fail. The program checks them before those commands are processed, and also catches and bypasses some errors and ensures smooth operation.

Problems Encountered and Their Solutions

The two biggest problems encountered during the writing of the program are the faulty situations that the program has to deal with and the complex time system of the program. The solution to the first problem is complicated because no incorrect input should be sent when creating objects representing devices. The reason for this requirement is that if an incorrect input is sent, it can affect other critical methods of the program until that corrupt object is deleted. For this reason, the correctness of the command is checked first and then each input is checked in turn. The solution of the second problem requires the time system and the ordering method to work in an integrated, harmonious manner. The complexity of the code is reduced by using a method that controls all devices in the system and constantly changes the device order when necessary in the commands where these two are used.

Benefits of the System

The system is basically designed to save people time and make life easier. The system performs many methods by itself, ensuring that people do not have to do the repeated processes that they need to do. At the same time, it prevents misuse of devices thanks to its error management. In this way, it is possible to extend the life of the devices.

Benefits of OOP

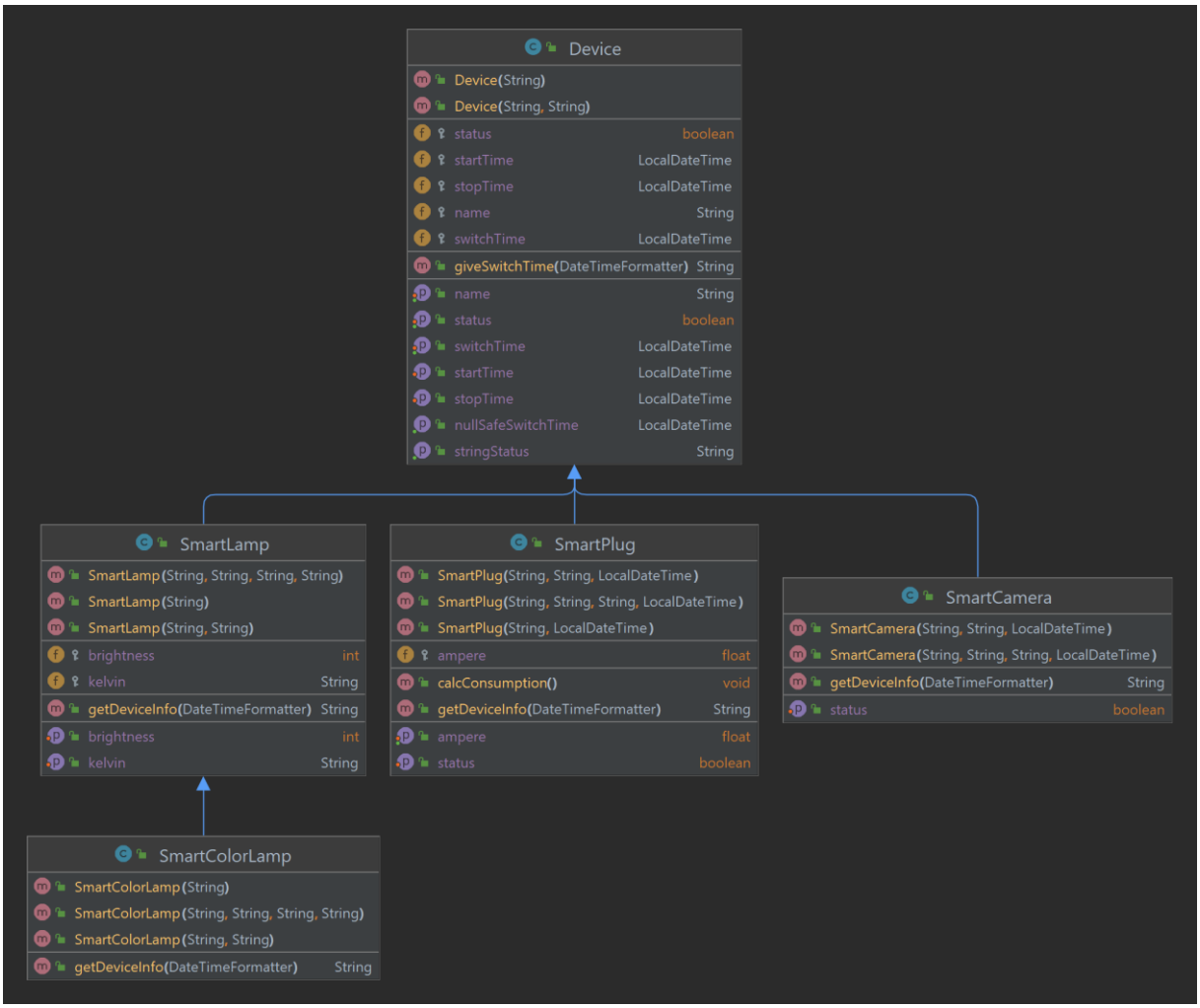
Object-oriented programming has made it possible to do such a complex project. Basically, managed things are devices, and although they have common features, they are actually unique to themselves. Inheritance and polymorphism pillars of object-oriented programming have prevented writing long code blocks each time and paved the way for easy use of methods suitable for those objects. OOP's

abstraction and encapsulation features enabled the device management part of the program and other methods running the program to be changed and developed independently. In this way, development time is shortened and code complexity is reduced.

Four Pillars of OOP

As mentioned in the previous section, the four pillars of OOP are abstraction, encapsulation, inheritance, and polymorphism. Abstraction is the separation and hiding of the bottom layer that does the job from the top layer that uses it. Encapsulation is a concept related to abstraction. Encapsulation is hiding the data used by the methods that do that job and providing access only by those methods. Inheritance is the ability of some classes to use properties and methods of higher types to which they belong. Polymorphism, on the other hand, is the ability to be used as if it has the type of the class above it, since the classes below the hierarchy also belong to the classes above it.

UML Diagram and Explanation of the UML Diagram



As mentioned in the previous sections, there are 5 classes in the program. The top class in the hierarchy contains the *Device* class and the properties and methods that all devices have. The other 3 devices are sub-classes of *Device* classes. The color smart lamp is a subclass of it because it has the features of the smart lamp. Getters and setters used in the main program are defined in all classes. In this way, the attributes of classes can be defined as protected and OOP's encapsulation pillar can be fully used. All classes have a *getDeviceInfo* method that returns their unique properties as a String. In this way, when information is requested from any device, information can be requested directly from

the class. *SmartPlug* and *SmartCamera* have a overridden method that do calculations in their status setters'. This is because both classes have a feature that requires computation. In this way, calculations can be made every time these devices are turned off and on.