Hacettepe University
Departmant Of Computer Engineeering

SMART HOME SYSTEM REPORT

Emirhan Utku-2210765029

19.04.2023

INDEX

1)Problem Definition

In this assignment, a good OOP design is expected to be implemented for controlling and managing three smart home accessories - Smart Lamp (with white-ambiance and color-white-ambiance variants), Smart Plug, and Smart Camera. The devices are assumed to be controlled and managed with respect to commands, and it is expected that they are always kept in sorted order according to their switch times in ascending order. If there is no switch time, it will be remained in the order it was added. The implementation of this project is expected to obey the four pillars of OOP - Inheritance, Polymorphism, Abstraction, and Encapsulation - with consideration given to the trade-offs between them. The aim of the project is to make life easier by efficiently and effectively managing these devices and their time.

2) My Solution Approach for Problem

The common features of the devices to be added to the system led me to first create a common class named SmartDevices, and inherit the specific SmartLamp, SmartPlug, and SmartCamera classes that I will create from this class. Of course, each device has its own specific features within these classes. Similarly, since most of the events related to these devices are common, I created a BaseManagement class and inherited the SmartLampManagement, SmartPlugManagement, and SmartCameraManagement classes from this class. Of course, each of these classes also contains separate methods specific to each device, such as add and getInfo.

3) Problems and Solutions Encountered

The energy consumption of the cameras and plugs was found to have some issues. In order to solve this problem, two different situations had to be considered. In the first situation, the current time was used when the device was first turned on, which was determined by commands such as Switch PlugIn, and a separate method was written for this purpose. In the second situation, when the initial time was changed with commands such as Skip, SetTime, Nop, the device's first turn-on time was set according to switch times, and a separate method was written for this. In this method, if the switch times of the devices were before the current time, the turn-on time was set to switch times.

To solve the issue of devices whose switch times became null because time passed after their creation and the need to overwrite them onto the devices whose switch times were null from the beginning, two array lists were used, one for devices whose switch times were not null and the other for devices whose switch times were null. The devices with null switch times were removed from the first array list and added to the beginning of the second array list.

4)Benefits Of System

The feature that is considered the most important aspect of this system is that much can be achieved with little effort. Any device can be plugged in or unplugged at any time as desired.

Due to forgetfulness being one of the biggest problems faced in our time, with this system, there will be no need to worry about whether a device was left plugged in or if the lights were turned off after leaving the house

As energy consumption of devices is expected to be one of the biggest issues the world will face in the future, with this system, the energy consumption of devices can be monitored and energy can be saved accordingly.

5)Benefits Of OOP

Difficulties can be easily located when working with object-oriented programming languages. "Oh, the car thing failed?" The Car class must be related to the problem!" There is no need to go through every word of the code line by line. Self-containment is a characteristic of objects, and each piece of functionality operates independently of the others. Additionally, modularity allows for several objects to be worked on by an IT team simultaneously, reducing the possibility of one person duplicating the capabilities of another.

OOP systems can be easily upgraded from small to large systems.

The use of current classes can be increased and duplicate code can be minimized by utilizing inheritance

Another benefit of OOP is the ability to have code encapsulated. Encapsulation involves hiding the properties and behavior of an object from outside code, which helps to prevent unintended consequences that may result from changes made to the object. Additionally, a cleaner and more organized code structure is promoted by encapsulation.

6)Four Pillars of OOP

Inheritance, Polymorphism, Encapsulation and Abstraction are the four pillars.

6.1) Abstraction

The process of selecting and representing only the relevant characteristics of an object or system is known as abstraction. By using abstraction, the essential features of an object or system can be focused on by programmers, and unnecessary details can be hidden from users.

6.2) Encapsulation

The process of enclosing data and methods within a class, such that they are hidden from the outside world and can only be accessed through the class's public interface, is known as encapsulation.

6.3) Inheritance

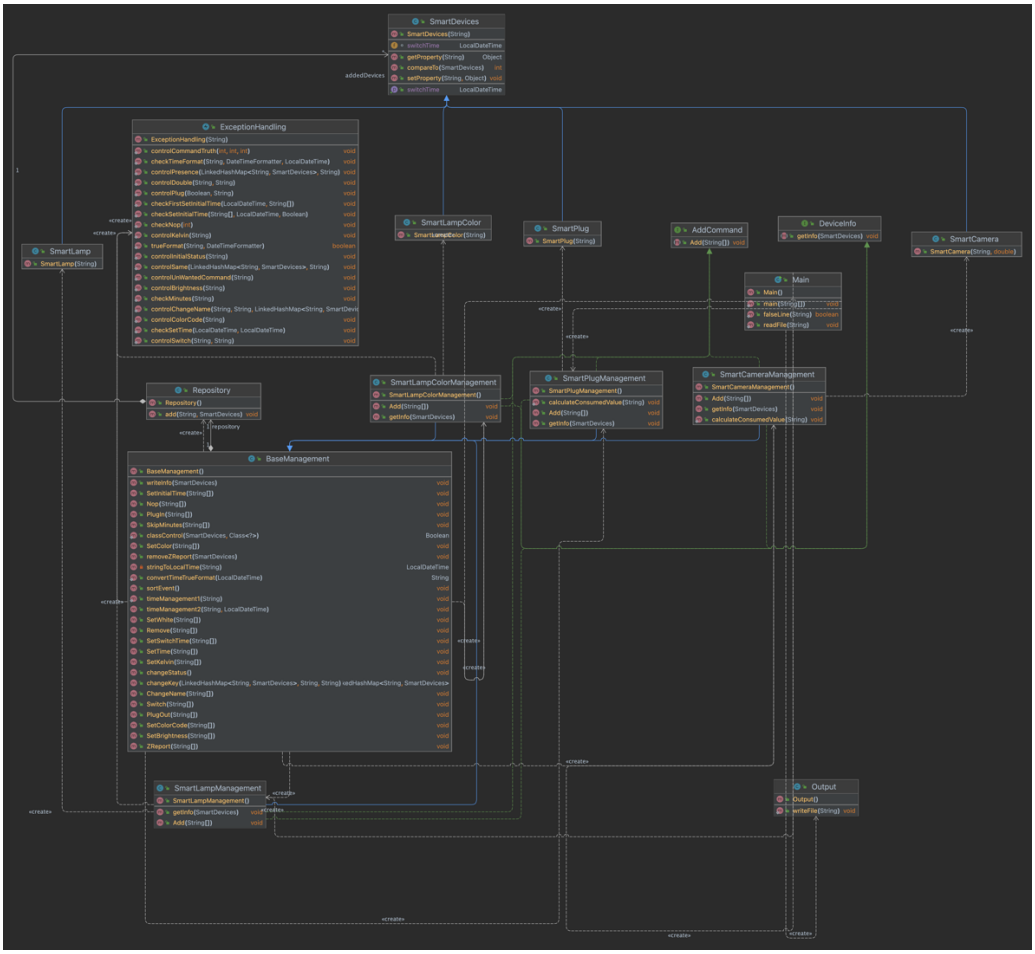
The mechanism by which a new class is derived from an existing class is known as inheritance. This allows the properties and methods of the parent class to be inherited by the new class.

6.4) Polymorphism

The ability of an object to take on multiple forms is known as polymorphism. In OOP, polymorphism is achieved through method overriding and method overloading.

A type of static structure diagram that describes the structure of a system is shown by a class diagram in the Unified Modeling Language (UML). The system's classes, their attributes, operations (or methods), and the relationships among objects are shown.

7)UML Diagram of My OOP Design



The SmartDevices class contains common properties for all devices and methods for getting/changing device properties. There are two interfaces, DeviceInfo and Add, used for writing device information to the system and adding new devices. Each specific Management class implements these interfaces.

The BaseManagement class contains common commands for all devices, and the Management classes for specific devices inherit from this class.

The Repository class is called statically from within the BaseManagement class, allowing all devices to be stored through this class.

The Output class is used within different classes for writing errors and writing device information to a file.