Hacettepe University

Computer Engineering Department

BBM104 – Assignment Report

Assignment 2

Smart Home System

Yusuf Demir

2210356074

Ankara

April/2023

Contents

1.Problem Definition and Solution

1.1 Description of the System:

The system consists of various smart home accessories, such as Smart Lamp (with white-ambiance and color-white-ambiance variants), Smart Plug, and Smart Camera. The system controls these devices and time with respect to commands. The devices must always be in sorted order according to their switch times in ascending order.

1.2 Solution Approach

The problem was approached by utilizing OOP principles to design and implement the smart home system. Three classes were created for the three smart home accessories - Smart Lamp, Smart Plug, and Smart Camera - with each class having its own properties such as kelvin, mps, and ampere. Additionally, a SmartDevices class was created that served as the parent class for all smart home accessories and contained common properties such as deviceName and switchOnTime. The four pillars of OOP - Inheritance, Polymorphism, Abstraction, and Encapsulation - were implemented to ensure that the system is easily maintainable and expandable in the future.

1.3 Problems and Solutions

The first problem that was faced, the problem was caused by a device having more than one "Add" command. I conveniently solved this problem by creating multiple constructors, which is one of the good features of "OOP". The only difference between the constructors was the different variables entered in the "Add" command. Furthermore, since SmartColorLamp is a child class of SmartLamp, the super keyword was used instead of rewriting some constructors. Thus, inheritance and polymorphism features were also utilized. The second problem that was faced, the devices must be sorted in ascending order according to their switch times, and exceptional situations where a device does not have any switch time should be handled by the system. In an attempt to solve the problem, the hashmap containing the devices was sorted. However, sorting the hashmap according to the desired shape was not successful. During the research process, it was discovered that a list can be sorted in the desired way. From the desire to continue using the convenience of Hashmap, the solution was found by leveraging both data structures.

2.About This System

2.1 Benefits of This System

The smart home system that was designed and implemented provides numerous benefits to users, including automation and convenience. The smart home accessories can be controlled through commands, and the devices can be sorted in ascending order based on their switch times. This system enables users to schedule their devices to turn on/off at specific times, which can result in energy savings and reduced electricity bills. Also by following the four pillars of OOP, the system is designed to be scalable and extensible, allowing new types of smart home accessories to be easily added in the future.

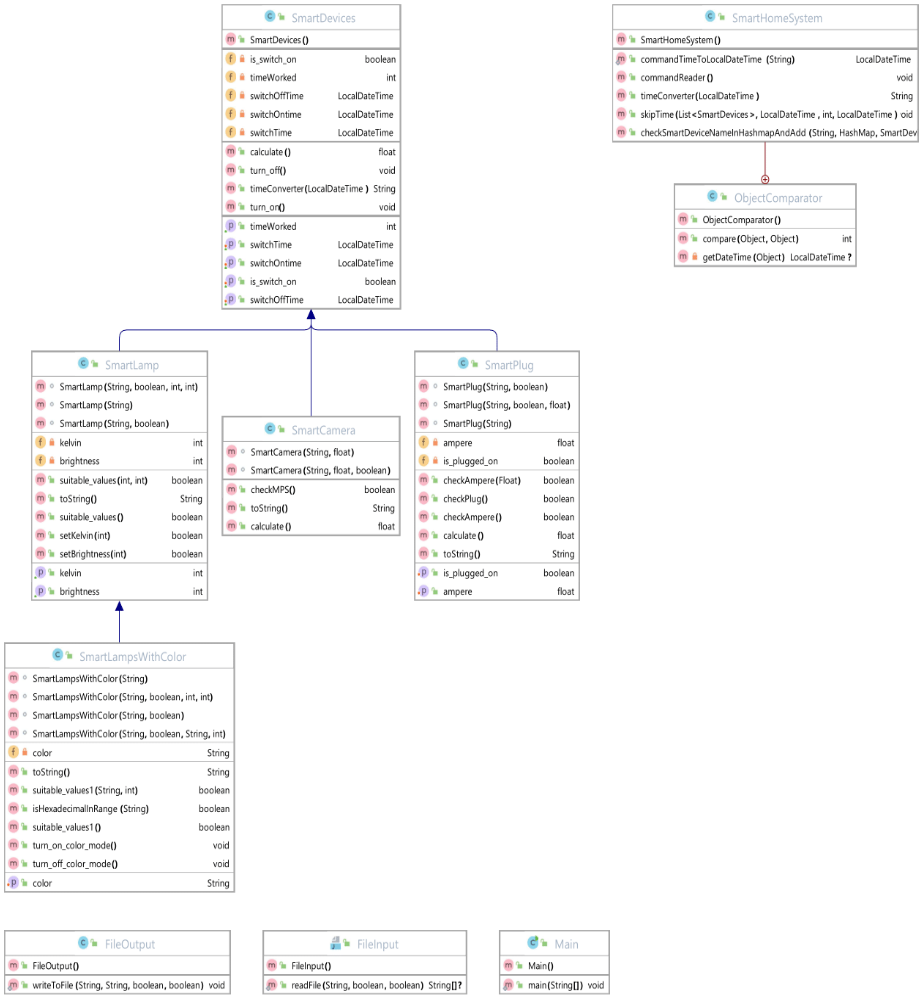
2.2 Benefits of OOP

Object-Oriented Programming (OOP) has numerous benefits, such as increased code reusability, improved software maintainability, and more efficient code organization. OOP allows developers to reuse code easily, saving time and reducing the amount of code that needs to be written. It also improves software maintainability by making it easier to modify code without affecting other parts of the program. Furthermore, OOP provides a more efficient way to organize code, making it easier to understand and debug. The four pillars of OOP are Inheritance, Polymorphism, Abstraction, and

Encapsulation. Inheritance allows classes to inherit properties and methods from other classes, making it easier to reuse code. Polymorphism allows objects to take on different forms, making it possible to use them in different contexts. Abstraction allows developers to hide unnecessary details and focus on the essential features of a program. Encapsulation allows developers to control access to certain parts of the program.

3.UML Diagram

UML is a standardized modeling language used to visualize and design software systems. UML diagrams can be used to represent the various components of a software system, including classes, objects, and their relationships. By using UML, developers can create a visual representation of the system, promoting easier understanding and communication between team members.


SmartDevices
SmartDevices()
is_switch_on boolean
timeWorked int
switchOffTime LocalDateTime
switchOntime LocalDateTime
switchTime LocalDateTime
calculate() float
turn_off() void
timeConverter(LocalDateTime) String
turn_on() void
timeWorked int
switchTime LocalDateTime
switchOntime LocalDateTime
is_switch_on boolean
switchOffTime LocalDateTime
SmartHomeSystem
SmartHomeSystem()
commandTimeToLocalDateTime (String) LocalDateTime
commandReader() void
timeConverter(LocalDateTime) String
skipTime(List<SmartDevices>, LocalDateTime , int, LocalDateTime) oid
checkSmartDeviceNameInHashmapAndAdd (String, HashMap, SmartDev
ObjectComparator
ObjectComparator()
compare(Object, Object) int
getDateTime (Object) LocalDateTime ?
SmartLamp
SmartLamp(String, boolean, int, int)
SmartLamp(String)
SmartLamp(String, boolean)
kelvin int
brightness int
suitable_values(int, int) boolean
toString() String
suitable_values() boolean
setKelvin(int) boolean
setBrightness(int) boolean
kelvin int
brightness int
SmartCamera
SmartCamera(String, float)
SmartCamera(String, float, boolean)
checkMPS() boolean
toString() String
calculate() float
SmartPlug
SmartPlug(String, boolean)
SmartPlug(String, boolean, float)
SmartPlug(String)
ampere float
is_plugged_on boolean
checkAmpere(Float) boolean
checkPlug() boolean
checkAmpere() boolean
calculate() float
toString() String
is_plugged_on boolean
ampere float
SmartLampsWithColor
SmartLampsWithColor(String)
SmartLampsWithColor(String, boolean, int, int)
SmartLampsWithColor(String, boolean)
SmartLampsWithColor(String, boolean, String, int)
color String
toString() String
suitable_values1 (String, int) boolean
isHexadecimalInRange (String) boolean
suitable_values1 () boolean
turn_on_color_mode() void
turn_off_color_mode() void
color String
FileOutput
FileOutput()
writeToFile (String, String, boolean, boolean) void
FileInput
FileInput()
readFile (String, boolean, boolean) String[]?
Main
Main()
main(String[]) void

The UML diagram shows three primary device classes: SmartLamp, SmartCamera, and SmartPlug. These classes share some common features and are thus grouped under a parent class called SmartDevices, which contains common properties and methods. Furthermore, the SmartColorLamp child class is nested under SmartLamp, inheriting its features and including a color mode. The SmartHomeSystem class handles most operations, including reading code, making time arrangements, sorting and editing lists and hashmaps, with the exception of smart device classes.