T.C.

HACETTEPE UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

2023 SPRING TERM

BBM 104 Assignment 2 Report

21/04/2023

Gülvera Yazılıtaş – 2210356111

INDEX

1 Problem Definition

Home devices are used every day by all people. As technology improved, household appliances also diversified. Because of that controlling every device every day became harder for humankind. Complex systems started to need simple control systems. So designing and implementing smart and inclusive home devices system is inevitable. The problem is creating this software system optimally and simply for users. This smart home system ought to control four main components, namely, Smart Plug, Smart Camera, Smart Lamp, and Smart Color Lamp.

2 Solution Approach

The solution approach for this system is creating components of devices abstract from each other but also gathering them in one whole program. Their functions should work separately and give their outputs together at the same time. For this kind of program, object-oriented design suits it the best. Starting to solve the problem from this point of view would contribute to the efficiency of the system. Thinking of every device as an object and creating them from instances of the proper class would help modularity and encapsulate the objects with their attributes and methods.

3 Problems Was Faced and Their Solutions

The Accessing and Calling the Objects Problem:

Since the program has to hold every object without number limitation, gathering them in ArrayListes was needed. However, despite every object having unique names, accessing them by their names was a problem. At this point two possible solutions may be offered: one is checking names by putting the objects list in a for loop and finding the index of the object that has enquired name; the other one is creating two lists one for objects and one for their name do that index of that object can be found easily by checking the name list. In this project second solution was chosen.

The Checking the Errors and Exceptions Problem:

This program takes its command from the user. This paves the way for getting wrong format inputs that may be caused to terminate the program. So a whole new errors class is defined in the program so that every possible exception can be checked and the program prints out the proper message to the user. The solution was aimed to be abstracted from the reading and carry out the commands class. So it became easier to add new error and exception definitions or remove the unnecessary ones without changing anything with the reading commands class.

The Problem of Sorting of the Reports of the Objects at Z Report :

When giving the z report program should sort them according to their switch times and if they do not have one their order should be in the same order as the time they have created. The objects that become null after the switch operation pose a sorting problem. The solution to this was thought like that; if any object becomes null it is added to the beginning of the null switch time objects list.

4 Benefits of The System

This system offers a lot of benefits. First off controlling lamps, plugs, and cameras provide convenience. People do not have to be at their houses to switch their devices and make arrangements. The second benefit is energy saving which is vital for the earth. People can avoid unnecessary energy consumption by setting switch times on their devices. The last but not the least benefit is security. Smart cameras can record important incidents. Also forgetting devices on plugs is no longer angst. Plugs can be switched off remotely.

5 Benefits of OOP

OOP is quite useful for designing efficient programs. It encapsulates objects data and methods which provides reusability and modularity. Creating classes and inheriting new ones from one another makes it easier to add new functions to the program and remove unused ones. So that programs that are designed with oop can live longer without changing the main parts of the design.

6 The Four Pillars of Object-Oriented Programming

1 Abstraction: It focuses on what objects do, not how does it. It hides the details of the process from the user and just shows essential info. Creating abstract classes and interfaces may be given as examples of the application of abstraction.

2 Encapsulation: It encapsulates the data and methods of an object. This prevents unauthorized access and modification. Access modifiers can be used when encapsulation. It is a good programming exercise that defines attributes of a class private and defines methods public.

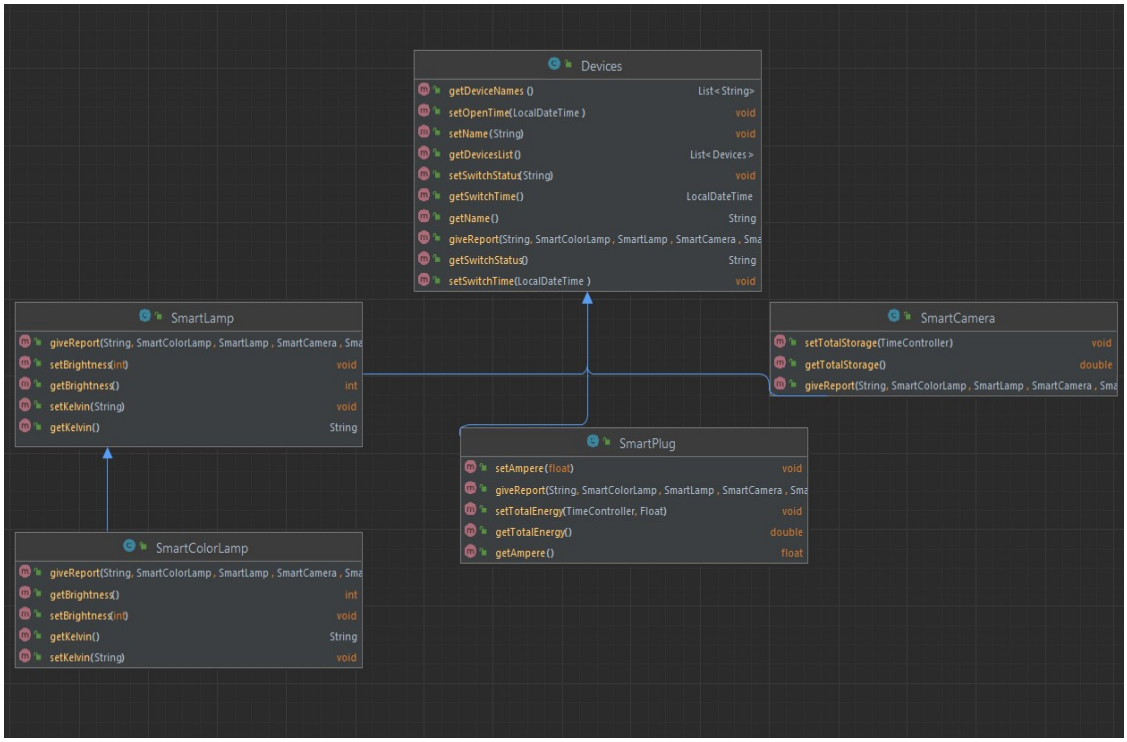
3 Inheritance: Creating new classes from existing classes. It enables a new class to inherit the data and methods of an existing class. Besides its superclass's data, and the method it can create its own attributes and methods. Inheritance means "is a" relationship between the classes

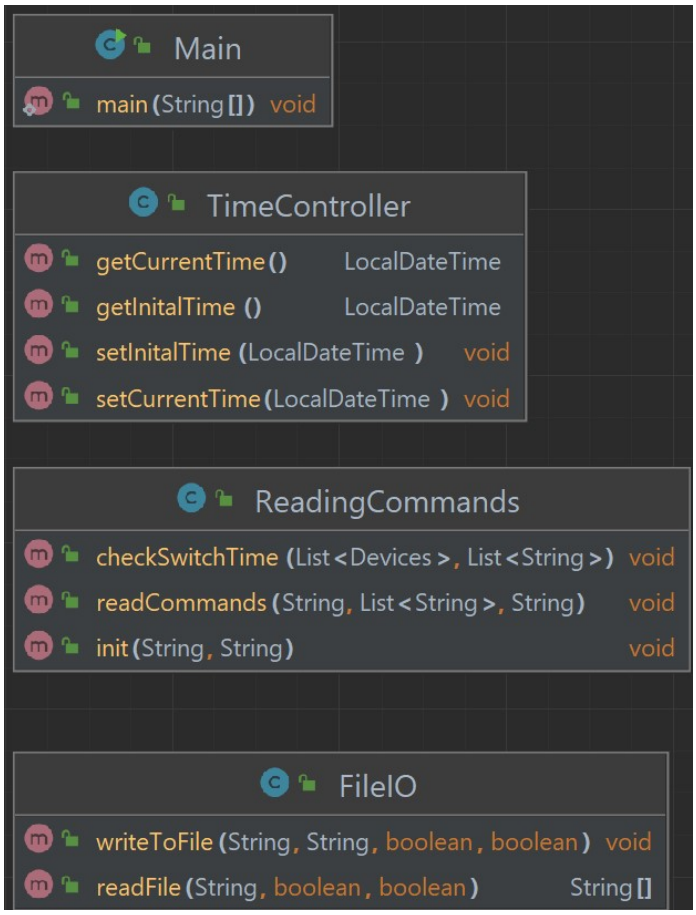
4 Polymorphism: An object can be created in many forms by referencing its superclasses. Polymorphism is achieved by using inheritance and interfaces. And its applications are method overloading and overriding.

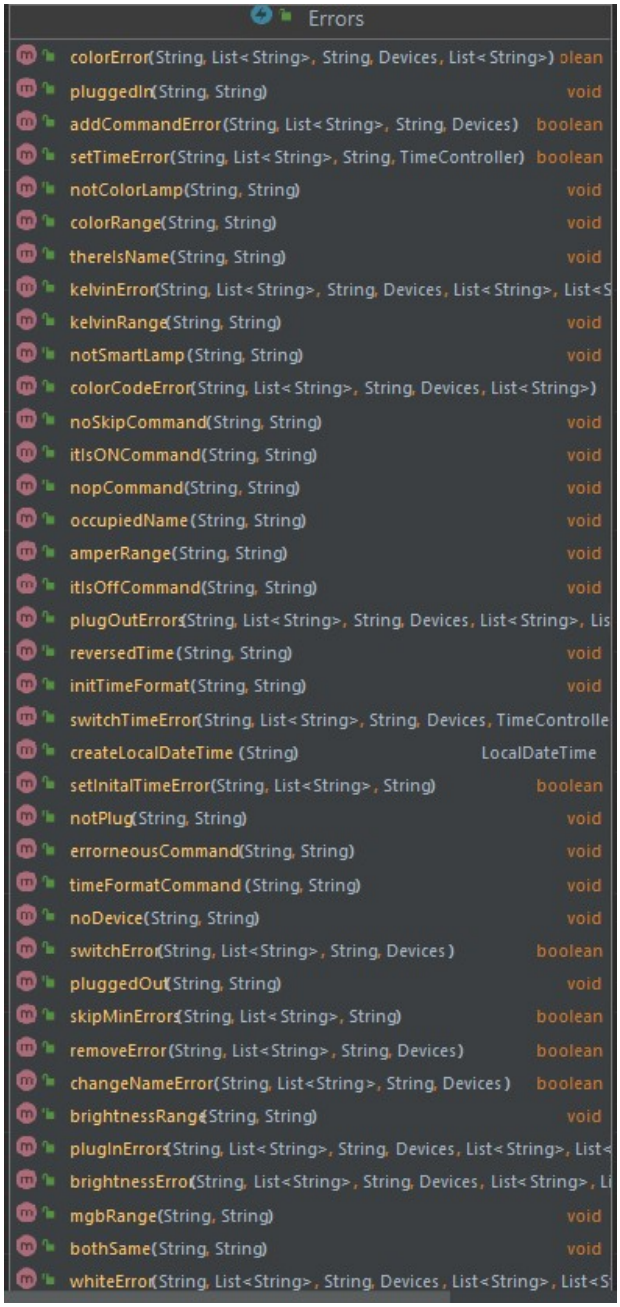
7 UML Diagram and Its Expanation

Unified Modeling Language (UML) is a graphical language used to show classes and their attributes and methods also relationships with each other.

The below uml diagram shows this program. As can be seen, there is one Main class that just makes code fragments run, one *TimeController* class for checking the current time, one *FileIO* class that is used for reading input from text files and writing output to them, one *Errors* class for checking all exceptions for all commands and write their messages(which are also created as methods that is why it is seen to much), one *ReadingCommands* for initialise program and execute all commands.There is one *Devices* super class that is extended by three sub-classes. One of these sub-classes, *SmartLamp*, is also a superclass of *Smart Color Lamp.*


Devices
getDeviceNames () List< String>
setOpenTime(LocalDateTime) void
setName(String) void
getDevicesList() List< Devices >
setSwitchStatus(String) void
getSwitchTime() LocalDateTime
getName() String
giveReport(String, SmartColorLamp , SmartLamp , SmartCamera , Sma
getSwitchStatus() String
setSwitchTime(LocalDateTime) void
SmartLamp
giveReport(String, SmartColorLamp , SmartLamp , SmartCamera , Sma
setBrightness(int) void
getBrightness() int
setKelvin(String) void
getKelvin() String
SmartCamera
setTotalStorage(TimeController) void
getTotalStorage() double
giveReport(String, SmartColorLamp , SmartLamp , SmartCamera , Sma
SmartPlug
setAmpere(float) void
giveReport(String, SmartColorLamp , SmartLamp , SmartCamera , Sma
setTotalEnergy(TimeController, Float) void
getTotalEnergy() double
getAmpere() float
SmartColorLamp
giveReport(String, SmartColorLamp , SmartLamp , SmartCamera , Sma
getBrightness() int
setBrightness(int) void
getKelvin() String
setKelvin(String) void


Main
main (String []) void
TimeController
getCurrentTime () LocalDateTime
getInitalTime () LocalDateTime
setInitalTime (LocalDateTime) void
setCurrentTime (LocalDateTime) void
ReadingCommands
checkSwitchTime (List < Devices >, List < String >) void
readCommands (String, List < String >, String) void
init (String, String) void
FileIO
writeToFile (String, String, boolean , boolean) void
readFile (String, boolean , boolean) String []


Errors
colorError(String, List< String>, String, Devices, List< String>) olean
pluggedIn(String, String) void
addCommandError(String, List< String>, String, Devices) boolean
setTimeError(String, List< String>, String, TimeController) boolean
notColorLamp(String, String) void
colorRange(String, String) void
thereIsName(String, String) void
kelvinError(String, List< String>, String, Devices, List< String>, List<S
kelvinRange(String, String) void
notSmartLamp(String, String) void
colorCodeError(String, List< String>, String, Devices, List< String>)
noSkipCommand(String, String) void
itIsONCommand(String, String) void
nopCommand(String, String) void
occupiedName(String, String) void
amperRange(String, String) void
itIsOffCommand(String, String) void
plugOutErrors(String, List< String>, String, Devices, List< String>, Lis
reversedTime(String, String) void
initTimeFormat(String, String) void
switchTimeError(String, List< String>, String, Devices, TimeControlle
createLocalDateTime(String) LocalDateTime
setInitalTimeError(String, List<String>, String) boolean
notPlug(String, String) void
errorneousCommand(String, String) void
timeFormatCommand(String, String) void
noDevice(String, String) void
switchError(String, List<String>, String, Devices) boolean
pluggedOut(String, String) void
skipMinErrors(String, List< String>, String) boolean
removeError(String, List<String>, String, Devices) boolean
changeNameError(String, List<String>, String, Devices) boolean
brightnessRange(String, String) void
plugInErrors(String, List< String>, String, Devices, List< String>, List<
brightnessError(String, List<String>, String, Devices, List< String>, Li
mgbRange(String, String) void
bothSame(String, String) void
whiteError(String, List<String>, String, Devices, List<String>, List<S