HACETTEPE UNIVERSITY
Department of Computer Science

Course: BBM103 Fall 2022

Assignment 2: Smart Home System

Mustafa Ege
2210356088

20.04.2023

INDEX

1.**Defining the Problem**

In this assignment it is expected from us to design a Java program for a smart home system that includes four smart devices and keep track of the time and control it for some features of these devices.

The way program works is that it takes a text file which contains all the commands as an input, then the program does the required process, finally it prints out all the commands and their results to another text file as the output.

There are 2 main problems in this system. First one is managing the time and the second one is implementing all the smart devices. These two problems are related with each other, since smart devices use the time to do some of their commands.

Starting with time, it is required to set an initial time for the system to start working properly, otherwise the system cannot do anything else even if there are other commands. Time can be changed after it has been initialized, by set time or skip minutes commands. Time input must be in the required form for each command and time cannot be reversed in any situation.

The first thing that connects time and the devices is setting switch times for devices. Devices has switched on or off properties. One way to change their status is by switch command, but it can be changed with time too. When there is the set switch time command, devices must hold the time in the command and switch the status when the time comes. The time can come with set time command, skip minutes command or nop command which skips the time forward to the earliest switch time.

All the smart devices have initial name and status. Name can be changed with change name command and status can be changed with switch command or if it has a switch time it can be changed with time. Smart devices are added by add command and can be removed with the remove command.

Smart plug is a device that calculates the energy consumption on itself. It has properties like ampere value, default voltage value, being switched on or off and having a device plugged in to it or out. When it is plugged in and switched on, it starts calculating the energy consumption and when it is plugged out or switched off, it stores the total consumed value. However, until this device is switched off or plugged out, it doesn't update the total energy consumption.

Smart camera is a device that can tell how much storage it uses. The way it works is quite similar to smart plug. It has additional megabytes consumed per record property. When it is switched on it starts calculating and when it is switched off it updates the storage it uses.

Smart lamp is a lamp which has adjustable brightness and kelvin values. These properties can be changed in a specified range. (2000K-6500K, 0-100%). Like other devices lamp can be switched on or off.

Smart color lamp is an advanced smart lamp. It has all the properties that smart lamp has, additionally it has the color mode. In color mode kelvin value is not important and it uses the color code value. Color code has a range too which is 0x000000 – 0xFFFFFF. The properties of smart lamp and smart color lamp can be changed with set commands. (set kelvin-brightness-color code-white-color)

2.Explaining the Solution Approach

Starting from the input, the input file is read by the method in FileInput class and transferred into an ArrayList. This ArrayList containing all the commands line by line is going to be used in a method which calls other required methods. This method is manageCommands method in the class commandManager. This method gets every line of commands one by one and checks the first item of the line to call the required method to start a process. Also, some of the errors is being handled inside this method, such as time is not initialized as a first command.

The methods that are called in the commandManager class are all static methods. The reason for this is every command has different features and needs different error handling. Therefore, instead of getting the object and calling the method on it, these static methods get the command line as a whole and do all the things they need to do by getting the data from the command line. The objects are generally get by deviceGetter method which returns the object from the devices list, and instance method are used for these objects inside of the static method.

Starting with time class, there is a static time variable that is going to hold the time all throughout the code. Time methods is going to change this variable when necessary. There is the setTime method which is used for setting the time to the time which is specified as a parameter. This method is used throughout the code for SetInitialTime, setTime, also for the nop commands. skipMinutes command takes all the command line as a parameter and skips forward the time the required amount.

There are 4 different classes for 4 different smart devices. All of them extend from superclass SmartDevice which is also an abstract class. SmartDevice class contains all the methods and fields which are common among all the subclass smart devices. Fields are name, status and switchTime. They all have their getters and setters. Smart plug and camera has another superclass between smartDevice class and themselves which is DevicesCalculatingTimeInterval, because they both calculate some values by calculating time interval. Additionally, smart color lamp extends from smart lamp.

DeviceAdder method calls the method which creates the individual device objects in their subclasses. Every subclasses' deviceAdder method works differently because every smart device has different features. Subclasses' deviceAdder method calls the required constructor based on the properties number command line has. For instance, smart plugs deviceAdder method calls the constructor with only name when the command line has 3 properties, like Add-SmartPlug-Plug1. If there are 4 or 5 properties, it calls the constructor with 2 or 3 parameters respectively. After the subclass device object is successfully created, the method adds this object into an ArrayList which is going to contain all of the smart devices in one place. This ArrayList is public static to access from every class and located in the SmartDevice abstract class. For checking if a device exists in this list, there is another ArrayList which contains names of all the devices. DeviceAdders add the names of the object to this list whenever it adds to the allDevicesList.

Continuing from smart device class, there is the changeName command which changes device's name if it is appropriate and prints out the specified errors. Remove method removes the device from the device list if it exists and switches off the device before removing it. This makes smart plug to update its energy consumption and smart camera to update its storage it used. After these it prints out the report of the device that had been removed, if something's wrong it prints out the errors specified. Both change name and remove methods gets the specific object with the help of deviceReturner.

Switch device method switches the device to the asked status if it is not already in that status and prints out the specified errors. Also, after switching it updates calculations of smart plug and camera by checking if the device that has been called is an instance of smart plug or camera. It also starts the time that they start calculatin if it is switched on by calling the method from their superclass.

SetSwitchTime method takes the time input and checks if the time is after or at the same time of the command's time. If so, it sets the swichTime field of the object, also it add this object to an ArrayList which contains all the devices that has a switch time value. This list is going to be used for checking if the time for those device has come, to take next time value for nop command and for printing out the devices with switch time in the first place when zReporting. If the device already exists in that list, it doesn't add it, since changing the switch time value will update the object in list as well.

DeviceSwitcherWithTime command is not a method that takes command line as an input unlike previous methods. This method is called whenever time is changed, so it is called inside the setTime and skipMinutes methods. But also it is called inside setSwichTime method, since switch time might have been set to the current time, which makes the device switch. It puts the items of the devicesWithSwitchTimes into a for loop and check their switchTimes, if they are before or at the same time of the current time, it switches the device off or on. It does the required things for smart plug and camera like switch method. Finally it calls the switchTimeCleaner.

SwitchTimeCleaner is for updating objects switchTimes to null after the time of their switch has passed, but it also makes the order for zReport for later use. Firstly, it adds all objects which used their switchtimes to a list. Then it removes all these objects from devicesWithSwitchTime list, finally in a for loop, it uses an algorithm to rearrange the allSmartDevices list by removing the objects from that list and adding it back to the first place and get the wanted order.

Nop command skips forward the time to the first object's switch time in the list by sorting it first. Then calls the switchTimeCleaner to remove the items that used their switch times.

There is an abstract report method which all subclass devices implement from. When Zreport method is called subclasses calls their own method to print out the required message. For the order zreport firstly prints the list with switch times then all devices list without the objects with switchtimes.

Smart plug, camera, lamp and smart lamp have their own private fields, setters, getters and methods for their specific features inside their classes.

2.1 Some of the Problems Faced and Solutions Found

- Too many specific errors to handle – user defined exceptions
- ZReport device order especially with same switch times – removing the device from the list and adding it back to the specified index with an algorithm.
- different causes of errors in the input command and having to print out the first error in add command – creating a testing object to find the error before adding to the list

3.Benefits of this System and OOP

This system separates the problem into many branches with the classes, so makes it much easier to build the whole system and solve the errors while working on it. Also, with the inheritance, instead of implementing every method to all classes individually, it is enough to having it in superclass and overriding it when necessary. Classes use private fields, so it keeps everything private and secure.

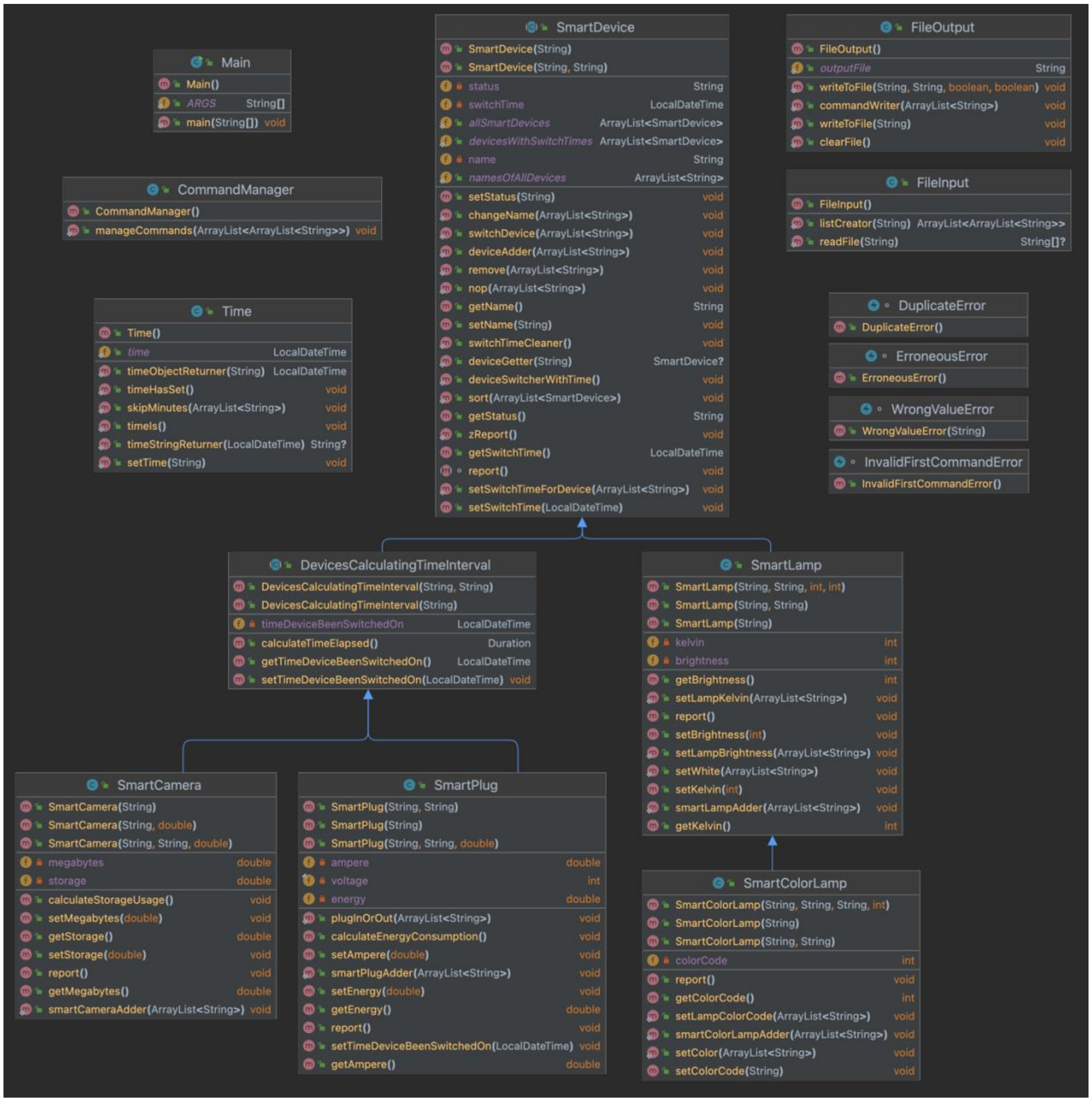
3.1 Four Pillars of OOP

- Encapsulation: Encapsulation stores attributes and methods in a single unit which is a class by using objects. Inside of the class is not directly accessible from the outside. It makes the code more organized, helps security and abstraction.
- Inheritance: Inheritance allows objects to inherit from super classes, which removes the need to use redundant code. It also allows subclasses to extend the features of super class.
- Polymorphism: Polymorphism allows different objects to implement the same method. It makes the code much more flexible to work with.
- Abstraction: Abstraction is for simplifying the complex system by using interfaces and abstract classes without showing the details of the code to see it simpler. Also, concrete classes can inherit from abstract classes or implement from interfaces to provide the details.

4.UML and This System's UML Diagram

UML is unified modelling language, it helps software developers to see the system's structure in a simpler way and understand it better. It includes diagrams with classes, fields, methods and others and showing the connection between them.

The UML diagram of this system includes 4 different classes for 4 different smart devices. Smart plug and smart camera have a mutual abstract parent class which is DevicesCalculatingTimeInterval class, since their mutual feature is that they calculate something with time. It contains the mutual methods and field that plug and camera uses. Smart color lamp extends from smart lamp, because it is a smart lamp with extra color mode. And these 2 classes devices calculating time interval and smart lamp extends from another abstract class SmartDevice. This SmartDevice class is actually the parent class of all 4 devices. It contains many methods and fields that all devices have in common. Other than these there are time class which deals with time, file input and output classes for I/O files, error classes, CommandManager class to call all the required methods and the Main class which contains the main method.



Main
Main()
ARGS String[]
main(String[]) void
CommandManager
CommandManager()
manageCommands(ArrayList<ArrayList<String>>) void
Time
Time()
time LocalDateTime
timeObjectReturner(String) LocalDateTime
timeHasSet() void
skipMinutes(ArrayList<String>) void
timeIs() void
timeStringReturner(LocalDateTime) String?
setTime(String) void
SmartDevice
SmartDevice(String)
SmartDevice(String, String)
status String
switchTime LocalDateTime
allSmartDevices ArrayList<SmartDevice>
devicesWithSwitchTimes ArrayList<SmartDevice>
name String
namesOfAllDevices ArrayList<String>
setStatus(String) void
changeName(ArrayList<String>) void
switchDevice(ArrayList<String>) void
deviceAdder(ArrayList<String>) void
remove(ArrayList<String>) void
nop(ArrayList<String>) void
getName() String
setName(String) void
switchTimeCleaner() void
deviceGetter(String) SmartDevice?
deviceSwitcherWithTime() void
sort(ArrayList<SmartDevice>) void
getStatus() String
zReport() void
getSwitchTime() LocalDateTime
report() void
setSwitchTimeForDevice(ArrayList<String>) void
setSwitchTime(LocalDateTime) void
FileOutput
FileOutput()
outputFile String
writeToFile(String, String, boolean, boolean) void
commandWriter(ArrayList<String>) void
writeToFile(String) void
clearFile() void
FileInput
FileInput()
listCreator(String) ArrayList<ArrayList<String>>
readFile(String) String[]?
DuplicateError
DuplicateError()
ErroneousError
ErroneousError()
WrongValueError
WrongValueError(String)
InvalidFirstCommandError
InvalidFirstCommandError()
DevicesCalculatingTimeInterval
DevicesCalculatingTimeInterval(String, String)
DevicesCalculatingTimeInterval(String)
timeDeviceBeenSwitchedOn LocalDateTime
calculateTimeElapsed() Duration
getTimeDeviceBeenSwitchedOn() LocalDateTime
setTimeDeviceBeenSwitchedOn(LocalDateTime) void
SmartLamp
SmartLamp(String, String, int, int)
SmartLamp(String, String)
SmartLamp(String)
kelvin int
brightness int
getBrightness() int
setLampKelvin(ArrayList<String>) void
report() void
setBrightness(int) void
setLampBrightness(ArrayList<String>) void
setWhite(ArrayList<String>) void
setKelvin(int) void
smartLampAdder(ArrayList<String>) void
getKelvin() int
SmartCamera
SmartCamera(String)
SmartCamera(String, double)
SmartCamera(String, String, double)
megabytes double
storage double
calculateStorageUsage() void
setMegabytes(double) void
getStorage() double
setStorage(double) void
report() void
getMegabytes() double
smartCameraAdder(ArrayList<String>) void
SmartPlug
SmartPlug(String, String)
SmartPlug(String)
SmartPlug(String, String, double)
ampere double
voltage int
energy double
plugInOrOut(ArrayList<String>) void
calculateEnergyConsumption() void
setAmpere(double) void
smartPlugAdder(ArrayList<String>) void
setEnergy(double) void
getEnergy() double
report() void
setTimeDeviceBeenSwitchedOn(LocalDateTime) void
getAmpere() double
SmartColorLamp
SmartColorLamp(String, String, String, int)
SmartColorLamp(String)
SmartColorLamp(String, String)
colorCode int
report() void
getColorCode() int
setLampColorCode(ArrayList<String>) void
smartColorLampAdder(ArrayList<String>) void
setColor(ArrayList<String>) void
setColorCode(String) void