HACETTEPE
ÜNİVERSİTESİ

CAN ERTAŞ - 2220356088

2ND TERM

ASSIGNMENT 2 REPORT

22/04/2023

Index

- Problem Definition

We quickly understand every smart device has at least three common features : the name of the device, the status (if it is on or off), the time when the status of the device will be switched (switch time). Besides every smart color lamp is a smart lamp. So I believed, polymorphism is a must for this assignment. Some classes needed more methods and variables such as SmartPlug and SmartCamera classes. These classes had more relationship with the time, so to update the used storage and consumed energy extra variables and methods were created in these classes.

The time concept of the assignment sounded even at the beginning like something I never experinced something similar before. Every time changing command like Nop, SetTime, SkipMinutes needed extra attention, I think. Because switching operations could have produced a big complexity. It was obvious that stored usage and energy consumption calculations would be hard to handle. By far, the sorting devices in the "*ZReport"* command required the most effort and made it necessary to write clean codes.

- Solution to Problem

I tried to code as clean as possible to understand and fix the errors in the output quickly. Thanks to OOP this was easier than I expected.

For every device, I created a class and these all were made the subclasses of SmartDevice class. This use of inheritance was essential in order not to create methods and variables that do the same operations, for multiple times. Then by using polymorphism, I was able to store my smart devices in one array which made everything easier.

Because the *"ZReport"* command requires knowledge of the switch events of the past. I created a variable to keep the time of when a command is given this variable is called currentTime and created also a critical variable in the "*SmartDevice* " class named lastSwitchTime which stores the last time of the switch operation executed due to "*SetSwitchTime"* command. When printing the ZReport the devices that has no set time where their status will be changed,

but has had their status changed before created complexity. Thanks to "***lastSwitchTime"*** variable this was fixed.

To reduce complex look and , I also used switch-case statements a lot.

- Benefits of This System and OOP

The analogy of classes being an idea of things in general and objects being a specific example of these things felt quite rigid, easier to handle problems and less time consuming. I could reach everything rigorously. I could connect classes between themselves way I want them connected. Every error or success can be handled class by class, java codes being composed individually different made pointing out things and redesigning them much easier.

- Four Pillars of OOP

Inheritance : inheritance is similar to the idea every human being a mammal and not every mammal being a human. We can say if a class has all the features of another class, then the first inherits the latter. The method overriding gives us the comfort to make methods more specialized for each subclass .

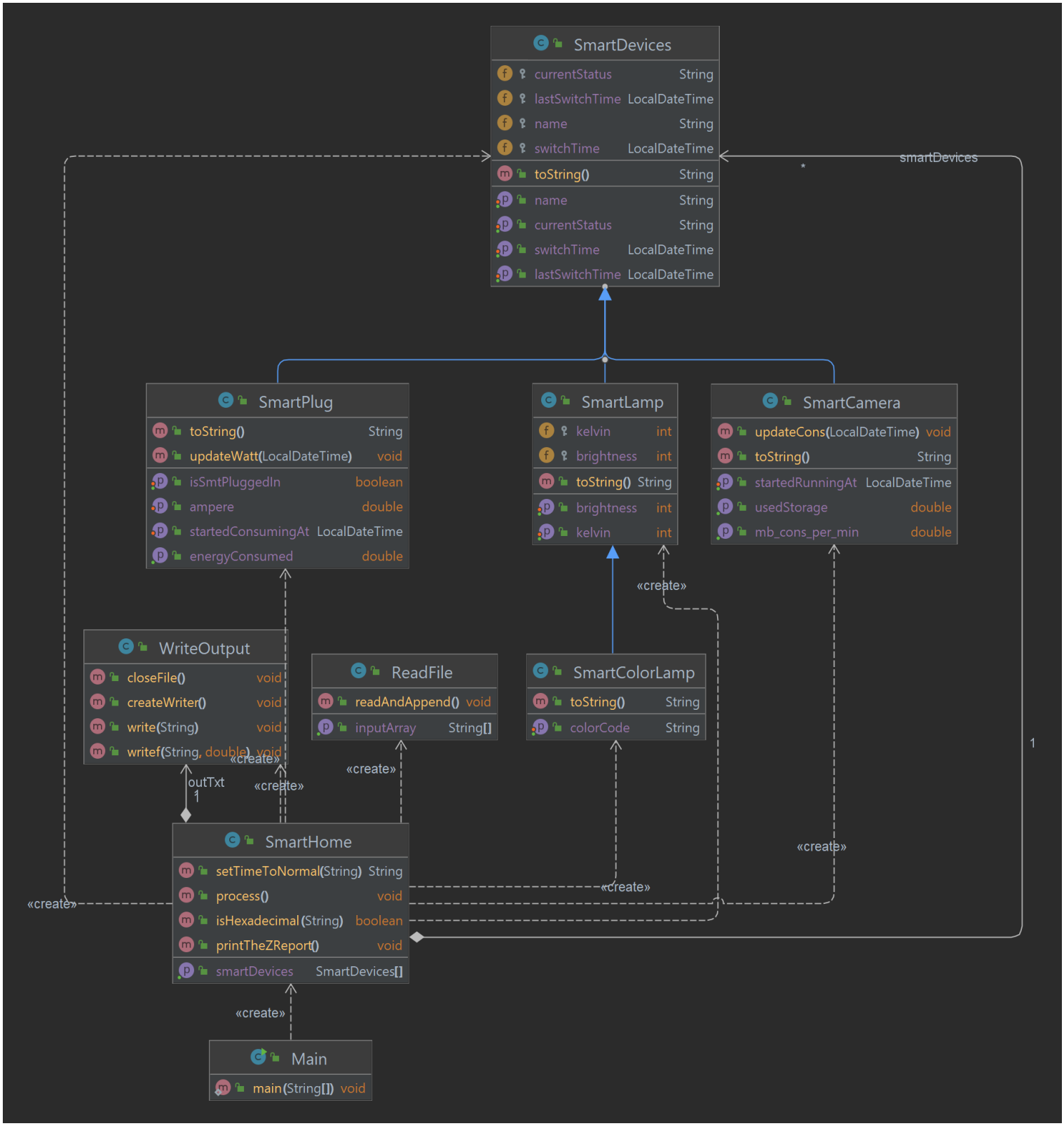
Encapsulation is simply data hiding. Unnecessary complexity and unwanted errors can be avoided easily by access modifiers. In the assignment, I kept the methods and fields private as much as possible.

Polymorphism, with the concept of inheritance lets us use the features of parent classes. Sometimes casting between classes is necessary to reach methods of child classes because "***objectName.method()***" does not always work because the method must also exist in the subclass.

Abstraction is the idea of a method or class, without details (implementation vs.). It represents of us students talking about future programs we want to design. We explain our ideas but not in code, we explain them "***abstractly"*** . Abstraction in java is something I admit failing to have gotten a strong understand of yet. I would like to study this more on the nextcoming assignments.

UML is for giving the user a visualized skeleton of the project to make it easier for them understand the project. It shows relationships between classes, show the fields and their data types and methods its parameters and parameters' types.

UML Diagram of my assignment:


SmartDevices
currentStatus String
lastSwitchTime LocalDateTime
name String
switchTime LocalDateTime
toString() String
name String
currentStatus String
switchTime LocalDateTime
lastSwitchTime LocalDateTime
smartDevices
*
1
SmartPlug
toString() String
updateWatt(LocalDateTime) void
isSmtPluggedIn boolean
ampere double
startedConsumingAt LocalDateTime
energyConsumed double
SmartLamp
kelvin int
brightness int
toString() String
brightness int
kelvin int
SmartCamera
updateCons(LocalDateTime) void
toString() String
startedRunningAt LocalDateTime
usedStorage double
mb_cons_per_min double
WriteOutput
closeFile() void
createWriter() void
write(String) void
writef(String, double) void
ReadFile
readAndAppend() void
inputArray String[]
SmartColorLamp
toString() String
colorCode String
SmartHome
setTimeToNormal(String) String
process() void
isHexadecimal(String) boolean
printTheZReport() void
smartDevices SmartDevices[]
Main
main(String[]) void
«create»
outTxt
1

- Resources

To understand how LocalDateTime class works, I used stackoverflow website. (https://stackoverflow.com/questions/22463062/how-can-i-parse-format-dates-with-localdatetime-java-8)