

PROJECT ASSIGNMENT 2

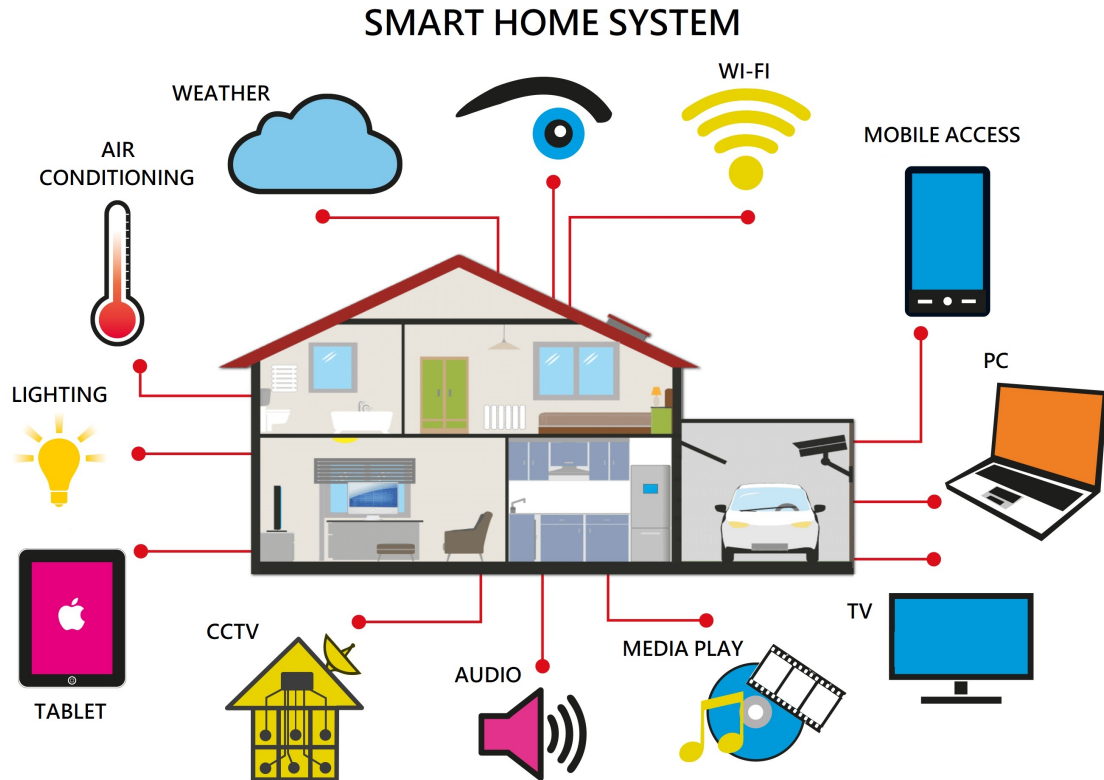
Issue Date : 31.03.2023 - Friday

Recitation Date : 31.03.2023 - Friday (14:00) (held on Zoom)

Due Date : 21.04.2022 - Friday (23:59:59)

Advisor : R.A. Görkem AKYILDIZ

Programming Language : Java 8 (Oracle)



1 Introduction

Humankind is a lazy thing indeed, according to our evolution progress, we always tried to do something with less and less effort. So, we invented many things to make our life easier, so that we can have more time to do "nothing". Automation is one of the things that we have done to feed this urge. So, smart home accessories is the one of the things that we have invented for the sake of automation. In short, we have done complex jobs to make our life easier, how ironic. In this project, you will be one of the people that tries to make our life easier for the sake of earning a bit more money, but with a "small" difference, unfortunately, you will not be paid for this project. So, enough with the story part, let's talk about the real things. OOP is what Java is based on, and it has very straightforward rules to obey. They are called as four pillars of OOP: Inheritance, Polymorphism, Abstraction, Encapsulation. If one obeys these four pillars with taking care of trade-offs, then it can be said that he/she knows what OOP is in basic. In this project, you are assumed to implement a good OOP design for given scenario and control the time flow.

2 Description of the System

In this project, there are some smart home accessories as follows: Smart Lamp (with white-ambiance and color-white-ambiance variants), Smart Plug, and Smart Camera. Variant of the accessories have been kept low to make scenario easy to understand and avoid any kind of confusions. The task here is, controlling these devices and time with respect to commands. Devices must be always in sorted order according to their switch times in ascending order. For exceptional situations, following rules will be applied: If there are two or more devices that have same switch time, their initial order (with respect to each other) will be preserved while sorting, if a device does not have any switch time, it will be considered as greater than all the devices. Note that, by default, each device is switched off and has no switch time at initialization if it is not stated otherwise. Moreover, switching the status of the device deletes switch time information.

2.1 Controlling the Time

This system is an autonomous system that obeys the given rules, the task must be done is getting the commands from the user and controlling the time when nop command is given. Description of the devices are as follows.

2.2 Smart Lamp

It is a lamp that its kelvin and brightness values can be adjusted in range of [2000K, 6500K] and [0%, 100%], they both are integers and their default values are 4000K and 100% respectively.

2.3 Smart Lamp with Color

It is same as smart lamp with an extra, it also has color mode, its color code is in range of [0x000000 0xFFFFFF], also color code is an integer but in hexadecimal base. It has two modes, color mode and white mode, if color mode is activated, kelvin value is not important, else, color code is not important. Default value for this lamp is as same as smart lamp (4000K, 100%).

2.4 Smart Plug

It is a wall plug that can calculate total energy consumption of itself. It is enough to update consumptions at events such as switching on/off, plugging in/out. Note that, if a smart plug is switched off, it does not mean that it is also plugged out. Default voltage value for it is 220 Volts. Ampere value is in positive number format. Moreover, initially there is nothing plugged in to the smart plug if it is not stated otherwise.

2.5 Smart Camera

It is a camera that give information about how much storage it uses. It is enough to update storage information at events such as switching on/off. Megabytes per minute is in non-negative number format.

3 Commands

Your program must give appropriate **error message** if it is tried to **adjust values out of the ranges**, it must give just the first error for each command. For example if it is tried to set white to 7000K, 150%, it must just give kelvin error. Note that, if provided kelvin or brightness values are not integer, as casting to integer is impossible at this situation, it will be considered as erroneous command. So, it is your own responsibility to check all kind of errors, such as if something must be an integer, then your program must give an error message about that if given value is not an integer. If something has to be in a range (such as second has to be in range of $0 \leq \text{SECOND} \leq 60$), then your program must give an error if it is not in the range as it has to be in.

3.1 Set Initial Time

This is the very first command that is given at input. Its mechanics are as follows:

`SetInitialTime <TAB> <YEAR>-<MONTH>-<DAY>_<HOUR>:<MINUTE>:<SECOND>`

Date will be in following format: Year: In full format, such as 2023. Month: **In integer format**, such as 3 or 03. Day: **In integer format**, such as 31, it can be padded for smaller days such as 3 as 03. Hour: In integer format, such as 14, it can be padded for smaller hours such as 3 as 03. Minutes: In integer format, such as 0 or 00. Seconds: In integer format, such as 0 or 00.

3.2 Set Time

Sets the time with respect to given time. **Does the required switch operations if there are any.**

`SetTime <TAB> <YEAR>-<MONTH>-<DAY>_<HOUR>:<MINUTE>:<SECOND>`

3.3 Skip Minutes

Minute can be any positive integer, **skips given amount of minutes**, same rules applied as set time command.

`SkipMinutes <TAB> <MINUTE>`

3.4 Nop

This command does nothing but **skips forwards the time to the first switch event and does the necessary switch operations**. **If there is not such an event, then it has to give an error. It is as follows:**

`Nop`

3.5 Adding Commands

For these command sets, program must add desired smart device to the system and print that command applied successfully unless it is an erroneous command. If there is an error at command, program must simply print an appropriate error message according to situation and it must discard that command for sure. Note that, names can be used as IDs, if a device with an existing name **is trying to be added**, your program must give an appropriate error message.

3.5.1 Adding a SmartPlug

This command's mechanics are as follows:

```
Add <TAB> SmartPlug <TAB> <NAME>
Add <TAB> SmartPlug <TAB> <NAME> <TAB> <INITIAL_STATUS>
Add <TAB> SmartPlug <TAB> <NAME> <TAB> <INITIAL_STATUS> <TAB> <AMPERE>
```

3.5.2 Adding a SmartCamera

This command's mechanics are as follows:

```
Add <TAB> SmartCamera <TAB> <NAME> <TAB> <MEGABYTES_CONSUMED_PER_RECORD>
Add <TAB> SmartCamera <TAB> <NAME> <TAB> <MEGABYTES_CONSUMED_PER_RECORD> <TAB> <INITIAL_STATUS>
```

3.5.3 Adding a SmartLamp

This command's mechanics are as follows:

```
Add <TAB> SmartLamp <TAB> <NAME>
Add <TAB> SmartLamp <TAB> <NAME> <TAB> <INITIAL_STATUS>
Add <TAB> SmartLamp <TAB> <NAME> <TAB> <INITIAL_STATUS> <TAB> <KELVIN_VALUE> <TAB> <BRIGHTNESS>
```

int?

3.5.4 Adding a SmartColorLamp

This command's mechanics are as follows:

```
Add <TAB> SmartColorLamp <TAB> <NAME>
Add <TAB> SmartColorLamp <TAB> <NAME> <TAB> <INITIAL_STATUS>
Add <TAB> SmartColorLamp <TAB> <NAME> <TAB> <INITIAL_STATUS> <TAB> <KELVIN_VALUE> <TAB> <BRIGHTNESS>
Add <TAB> SmartColorLamp <TAB> <NAME> <TAB> <INITIAL_STATUS> <TAB> <COLOR_CODE> <TAB> <BRIGHTNESS>
```

Note that, as both kelvin value and color code are integer, following trick can be used to discriminate them. Kelvin value will be given as decimal, such as 4000 whereas color code will be given as hexadecimal, such as 0xFA1D1F or 0x123456

3.6 Removing Commands

Remove <NAME> command will be used to remove a smart device from the system. If there is not such a device, system must give an error, else, it must print status of that device before removing. Note that, it has to switch the device off before removing it.

3.7 Determining a Switch Time for Device

Following command will be used for switching the device's status off or on. If there is not such a device, program must give appropriate error, else, it must give a message about result of set. Date format is as given before.

```
SetSwitchTime <TAB> <NAME> <TAB> <YEAR>-<MONTH>-<DAY>_<HOUR>:<MINUTE>:<SECOND>
```

3.8 Switching a Device to On or Off

Following command will be used for switching device to status of on or off. If there is not such a device, program must give appropriate error, else, it must give a message about result of switch, if it is already on or off, it must give an error. Smart cameras and smart plugs must update their total storage used and total energy consumption accordingly at switching of command.

```
Switch <TAB> <NAME> <TAB> <STATUS>
```

3.9 Changing name of a device

Following command will be used for changing name of a device, if there is not such a device, or new name is already exists, or there is no change at name, then your program must give appropriate error message.

```
ChangeName <TAB> <OLD_NAME> <TAB> <NEW_NAME>
```

3.10 Plug In Command for Smart Plug

Following command will be used for plugging in a device to smart plug, if that device is not a plug or there is not such a device or there is already a device plugged in or ampere value is not positive, then your program must give appropriate error. Note that, default voltage for smart plug is assumed as 220 Volts.

```
PlugIn <TAB> <NAME> <TAB> <AMPERE>
```

3.11 Plug Out Command for Smart Plug

Following command will be used for plugging out the device from smart plug, smart plug must update its total energy consumption when a plug is plugged out. Same errors at plug in command are valid for this scenario.

```
PlugOut <TAB> <NAME>
```

3.12 Set Kelvin, Brightness and Color Code

Following commands will be used to tune smart lamps' kelvin, brightness and color code values, if there is not such a device, or requested device is not a lamp, or not a color lamp (for color code part) program must give an appropriate error message.

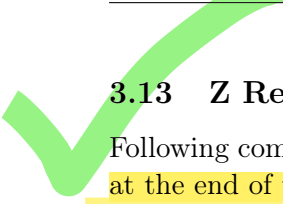
```
SetKelvin <TAB> <NAME> <TAB> <KELVIN>
```

```
SetBrightness <TAB> <NAME> <TAB> <BRIGHTNESS>
```

```
SetColorCode <TAB> <NAME> <TAB> <COLOR_CODE>
```

```
SetWhite <TAB> <NAME> <TAB> <KELVIN> <TAB> <BRIGHTNESS>
```

```
SetColor <TAB> <NAME> <TAB> <COLOR_CODE> <TAB> <BRIGHTNESS>
```



3.13 Z Report

Following command will be print Z Report of the system. This command will be runned once at the end of the command set even if it has not stated.

zReport



4 Report

You must prepare a report, the report must contain following parts:

- A cover
- An index
- The part that you define the problem
- The part that you explain your solution approach
- The part that you explain the problems you faced and the solutions you found to solve these problems
- Benefits of this system
- Benefits of OOP
- What are the four pillars of OOP (not just names of them but also their explanations in your own sentences) and UML?
- UML Diagram of your OOP Design and Explanation of the UML Diagram
- Resources (if you used any)
- Appendix (if you have any)
- The text at your report cannot be more than three pages and less than one page when it is copied to an empty Microsoft Word document with settings of font type Calibri, font size 11, so do not concern about images, cover and index pages as the limitation is only about the text that you write during your report. Your report must be in passive voice and it must be in justify alignment. Note that justify alignment is not needed to be obeyed for the headers or sections. Do not inject any code fragment to your report. Your reports must also be unique as your code, so do not share your code and report with anybody until objections are finalized.

5 Restrictions

- Your code must be able to compiled and executed on our department's developer server (dev.cs.hacettepe.edu.tr).
- You must obey given submit hierarchy and get score (1 point) from the submit system.
- Your code must be clean, do not forget that main method is just driver methods that means it is just for making your code fragments to run, not for using them as main

container, create classes and methods in necessary situations but use them as required. Moreover, use the four pillars of Object-Oriented Programming (Abstraction, Encapsulation, Inheritance, Polymorphism) if there is such a need, remember that your code must satisfy Object-Oriented Programming Principles, also you can benefit from exceptions and even if create your own exception class if you need any.

- You are encouraged to use lambda expressions which are introduced with **Java 8**, list structure of Java such as LinkedList, ArrayList etc., and your own exception classes.
- You must append a PDF file ("**Checklist.pdf**") in addition to your report that contains the requirements that you did not satisfy, please state your situation such as "This functionality does not work at all.", "There is a tiny issue such as ..." so that your assignment will be graded by taking them into consideration, please note that if you did not stated a requirement, that is assumed that that functionality is fully working on your system and that will be graded as zero even if a tiny issue occurs that stated as has to be checked (not for the issues that they are not stated at requirements, but remember that there may be some extra conditions that you should check, but they are not going to be graded as zero if you do not satisfy them). Moreover, if you did not provide such a file, it is assumed that your code is working well without issue and that it will be graded as zero even if a tiny issue occurs that stated as must be checked.
- You can benefit from Internet sources for inspiration but do not use any code that does not belong to you.
- You can discuss high-level (design) problems with your friends but do not share any code or implementation with anybody.
- You must use JavaDoc commenting style for this project, and you must give brief information about the challenging parts of your code, do not over comment as it is against clean code approach. Design your comments so that some wants to read your code can easily understand what is going on. You can check here to access Oracle's own guide about JavaDoc Sytle.
- Do not miss the submission deadline.
- Source code readability is a great of importance. Thus, write READABLE SOURCE CODE, comments, and clear MAIN function. This expectation will be graded as "clean code".
- Use UNDERSTANDABLE names to your variables, classes, and functions regardless of the length. The names of classes, attributes and methods should obey Java naming convention. This expectation will be graded as "coding standards".
- You can ask your questions through course's piazza group, and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but **DO NOT SHARE** answers, algorithms, source codes and reports.
- All assignments must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- Submit system for this homework will be opened a few days before deadline, so please be patient.

6 Execution and Test

Your code must be compiled and executed under **Java 8** and dev.cs.hacettepe.edu.tr. If your code does not compile and execute under developer server, then you will be graded as 0 for code part even if it works on your own machine. Sample run command is as follows:

Compilation: javac Main.java

Run: java Main input.txt output.txt

7 Grading

Task	Point
Implementing Smart Devices	25 (5 per each, 5 for full implementation)
Output (Including Errors)	30
Output	20
Comments in JavaDoc Style	10*
Report	15*
Total	100

* The score of your report will be multiplied by your overall score (excluding report and comments) divided by the maximum score that can be taken from these parts. Say that you got 60 from all parts excluding report and 25 from report, your score for report is going to be $25 \cdot (60/75)$ which is 20 and your overall score will be 80.

Note that, there will be two major overall multipliers, OOP Multiplier and Correct Submit Hierarchy Multiplier. If you do not obey the submit hierarchy (you have to score exactly 1 point at submit system, otherwise it will not be accepted as correct hierarchy even if your hierarchy fits to format), you will be penalized with 20% of point deduction. If you do not obey to OOP at all you will be graded as 0, and also there will be some values for OOP multiplier between 0 and 1 with respect to your design. So, OOP Design is as important as getting the correct output as you may get 0 from this project if you will not obey the OOP rules at all, even if your code outputs fully correct.

8 Submit Format

File hierarchy must be zipped before submitted (Not .rar, only not compressed .zip files because the system just supports .zip files).

- b<studentid>.zip
 - Report.pdf
 - Checklist.pdf
 - <src>
 - Main.java
 - *.java

9 Late Policy

You have two days for late submission. You will lose 10 points from maximum evaluation score for each day (your submitted study will be evaluated over 90 and 80 for each late submission day). You must submit your solution in at the most two days later than submission date, otherwise it will not be evaluated. Please do not e-mail to me even if you miss the deadline for a few seconds due to your own fault as it would be unfair for your friends, e-mail submissions will not be considered if you do not have a valid issue.