Technical report

Up2023682

Martin Dzhatov

I will begin my technical report with how I have complied with the constraints. The infrastructure of the code has been done considering the front end and the backend. I have separated my methods in their separate objects and also there are no functions which interact with the console inside of my backend objects. In my code I allow the user to have zero or many rooms, plugs and devices combinations. The code also uses mostly integer based input except where string input is more necessary. The code does not use any additional libraries nor does it have exception handling. I have done my best to not overcomplicate the code unnecessarily, there is no extra functionality.

I have started this project by using the legacy code from the worksheets.

Task 1

The ConsoleHelper was in development to the end of the project. I have kept most of the front end tasks inside the class. All methods that need to interact with the console either by receiving input or outputting data into the console have been created inside the Consolehelper.

Task 2

The dashboard was used only to build the initial objects and initiate the while loop which will keep the program running.

Task 4

The fourth task allows the program to initialize the “home” object giving the parameters of the input.

Task 5

This task uses the “Room” object to store the Name of the rooms and allocate the generated “id” by incrementing

Task 6

Task six is where the initial set up ends, here the code increments through “numberOfSmartPlugs” to assign the plug to the room and then the attached device to the plug itself.

Task 7

In this task the code is using the method “houseLevel” to go through all the plugs and either set their status to false or true.

Task 8

In task eight the code is using the “roomLevel” method which goes through all the plugs and checks if the room id will match with the one that the user has imputed. And then executes the code to the plugs inside the specific room.

Task 9

Task nine is similar to task eight but this time the code is using the “plugLevelOption” parameter to determine the pug id and either set the status to false or true.

Task 10

I found the best approach to task ten is to divide the switch statement into two separate statements, this is because I needed to output text to the console and this was the best solution without breaking the constraints. After printing the text the program asks the user for more input to proceed with functionality which is to change the attached device of a plug.

Task 11

Task eleven is part of the task ten’s switch, this is also a design choice because it requires more input from the user.

Task 12

The last task was very challenging, the only solution that I have found without breaking the constraints is creating a new object and then going through all the instances of the old object copying them to the new one. After that the new object will be copied on top of the original overwriting it. This allows the program to use the same object name for all the methods without the need of a new method for every object created. This however might be the result of why after the new object is created and added to the program, the program itself finishes and does not allow the console to execute the “menuOptions” method. The workaround also requires an empty constructor otherwise the program does not allow for an object without any variables. Also I have created a second method which populates the objects (plug,attached device, room) this is because when the users add an object the the populating method must start from where it last ended.

In general I have used the SmartHome class to store the functionality of the program and give all the objects their own responsibilities where needed.

Video:

https://www.youtube.com/watch?v=I29\_1YpqOj4