A Not So Smart, Smart-Home. PT.2

Guillaume Rousseau V00875834
Scott Hannan V00874232

11/16/2018

The second half of our solution included some design principles as well as some design trade-offs that were necessary for the development of our Smart Home system. We will be marked on all 36 .BDD tests.

TRADE OFFS:

-The main trade off we had was trading code readability in some places for functionality, obviously this is not ideal, however with such a large assignment we felt it was a necessary trade off.
-We traded functionality for looks, which was a trade off we deemed important as it is more important that the app has all functionality as required by the .BDD rather than be good looking.

PRINCIPLES WE USED:

-We decided it would be best to rid the program of the mediator interface as it did not add any benefit to the program's usability (The Interface Segregation Design Pattern). We thought the use of this had a positive effect on the coupling of the assignment.
-We aimed for loose coupling and achieved it in many places but some places we were unable
-Obviously we used the Observer Design Pattern with two models – Hub and Device.
-We were able to make use of threading to add a device status check at startup and at intervals, it did not interrupt the UI.

PRINCIPLES WE WOULD USE NEXT TIME:

-Sometimes it is good to look back on work and say where you would've changed the program to make it a more palatable piece of software.
-If we were to do this assignment again, we would've made use of the Iterator Design Pattern as lots of the functionality of the app has to do with iteration over many aggregations (Devices, Users). The benefit of using this would allow us to access elements of the aggregations without having to expose them so much.

WHAT WE TESTED FOR:

-Our tests were meant to be extensive, and we had hoped to achieve maximum path coverage.
-The UI tests we had were basic, however, we thought they had good effect in testing the basics of our UI.
-We did a lot of manual testing which we thought (for a system of this size) was the most effective as we were able to control the tests and see the output immediately and not have to mess with any potentially finicky tests.

CLASSES WE ADDED:

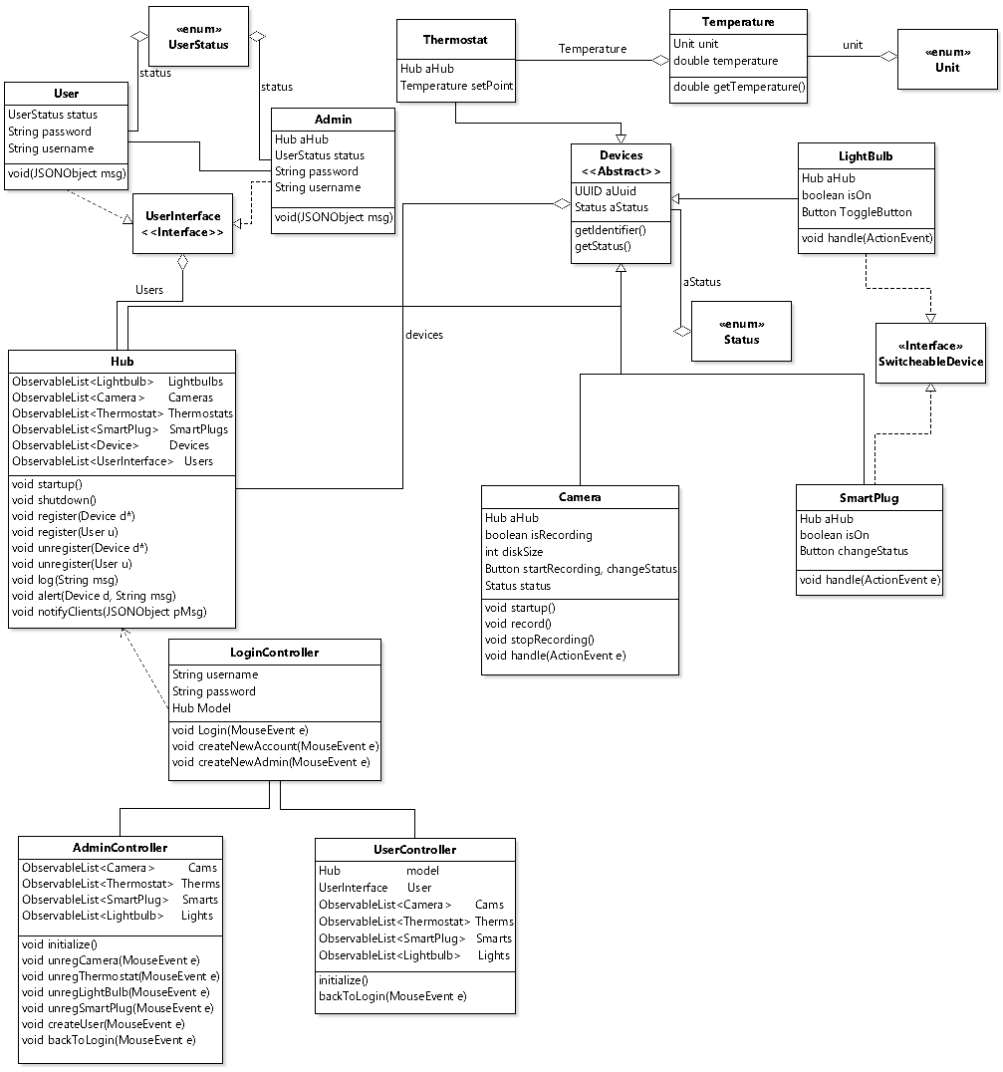
-Controllers (ADMIN/CONTROLLER)
-User Interface, an interface for users/admin.
-We didn't feel the need to add much as the base we were given was very comprehensive for our needs.

To run our program as an admin:
username: scott
password: admin

To run our program as a basic user:
username: gui
password: user

It is advisable to go back and forth between users in one instance of the program to see how changes affect things

To run our Test suite:
Might have to add the JRE-system Library 1.8
Run as a Gradle project (build.gradle)



A class diagram of the system, demonstrating the thickness of our main model, hub.