Kush Patel

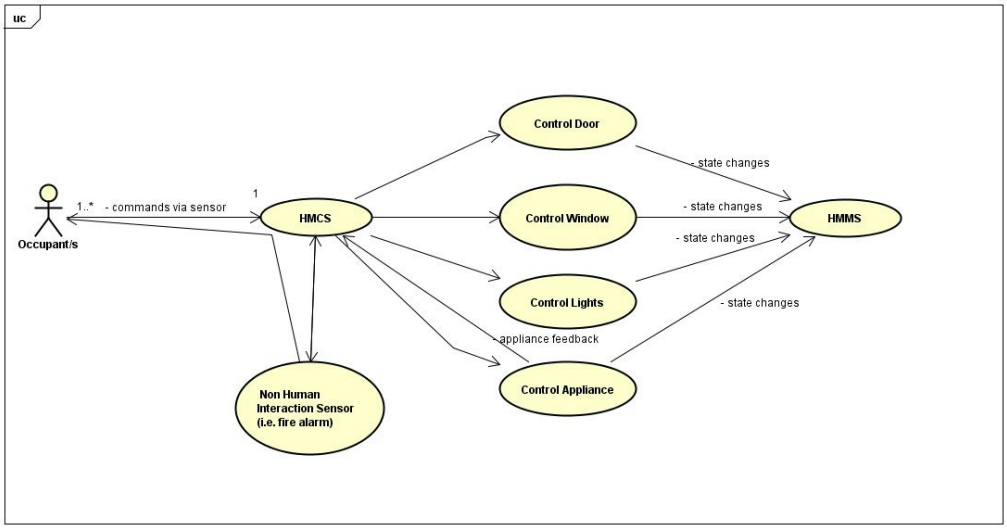
Assignment 3 Analysis

Some feedback I received:

- Use only 1 command type. Pass the execute method all the variables needed and make them generic. Pass nulls when that variable is not needed for the implementation.
- Make sure your use case diagram shows the voice interaction with the HMCS via the Ava appliances. Seperate the sensors out from 1 group to camera, smoke detector, and ava.
- You don't need to include all the methods and properties for the existing features. Just make a note to let the reader know to reference the existing documents and documents the changes here. It will make your class diagram smaller and tighter.

Changes from peer feedback and my own design analysis

- I redesigned the the class document and in doing so I didn't explain or include all the components in the House Mate system. I did include Occupant and Location specifically to show how I store the location for each occupant, update it, and retrieve it. House was included to show where the reset of the House Mate objects lie, as they all associate up to house.
- My original Use Case diagram, which also reflected on poor design grouped all the sensors and appliances together. This resulted in clustered methods and calls that set multiple nulls, all in order to reduce the number of methods used. Changing the use case and how the interactions occur, allowed me to change the design of the system while still maintaining the same results. (see below prior use case document)



- After changing the use case, I thought to use multiple commands; only seemed logical when the interactions were separate. However, because the command pattern allows for a very simple generic design, I was later able to combine the commands into a single form. However, this did not allow for me to use command pattern for the smoke detector and oven and other applianced. This development unfortunately got left out since my design altered from that path.

Design Document vs Implementation

During this assignment, the design document, through its many different phase, actually confused me during the implementation. Between changing the design, exploring different use cases, the result code changed through multiple phases of which many attempts failed to get the desired results. Also, because this system and requirements for this system originate from two prior designs, both from which my implementation deviated, it became much more difficult to develop the HMCS without changing many features such as location storage from the KG instead of the HMMS. Once I started following the path my implementations for the previous assignments took, I was able to get the results I desired even while deviating from the requirements.

Also, though it maybe be easier for someone completely familiar with UML and reading patterns from it, I found it very difficult to use the design doc to implement the pattern. The original design document had many more convoluted realization to the command and logically it made sense, but when put down into a UML class diagram, and then converted into code, it resulted in many failures. As my experience with different design patterns and UML increases, it will definitely improve this process, but as a beginner in both worlds it just increased confusion. Lots of information was lost or misconstrued when converting between mental thought, to a design pattern, to implementation.

As mentioned in the previous analysis, the peer design review is extremely helpful. Coming from a pair programming background, I've found peer input to be vital in developing robust solutions. They assisted me on fixing some problems in my design as well as clarifying the confusions I had. Also, while analyzing their implementation of the command pattern, I made some realizations of my own and understood it much more. I actually now appreciate this design and definitely want to implement it when an opportunity arises at work.