Housemate Entitlement Service Results

Erik Orlowski

Pattern Implementation

The assignment required four patterns to be used. The Visitor pattern, the Composite pattern, the Abstract Factory pattern and the Singleton pattern.

The Visitor pattern was a mixed bag. I think it was helpful, although not completely necessary in creating the inventory. This is a good example of where the pattern is helpful in iterating through objects. For checking access, I didn't like the fit of the visitor pattern. Fundamentally, I felt that checking access should start with a single, given, AccessToken and could use existing logic in the domain classes to determine entitlement matches. I think a solution utilizing the command pattern, or simply a custom solution likely would have yielded a cleaner and more elegant solution.

The Composite pattern was a perfect fit for the relationship between Roles and Resources and was very beneficial in access checking and entitlement management.

The Abstract Factory pattern also seemed like an awkward fit. While I don't believe implementing it hindered the design, it didn't really seem to fit any textbook use cases of the pattern and I don't believe it provided much value over using the class constructors directly.

Finally, the Singleton pattern was lightweight and was very helpful for the EntitlementServiceApi. A common problem I find is trying to get a reference to the object you're looking for, so when I can, I always like to use the Singleton (or even extending it to expose multiple concrete instances) pattern.

Improvements to Design

Most of my debugging time on the implementation was spent with String processing for commands. A lot of this processing wasn't included directly in the design and was left as an implementation problem. In the future, I think finding an elegant way to handle this String processing in the design and get to useable objects quicker would be beneficial.

Implementation Changes

In the initial design, the AccessToken was stored as a char array. This is mainly from some authentication sample code I saw when adding a dummy token in the Housemate Model assignment. In implementing this design, it proved difficult to compare this stored value against command line interface, so storing this value as a long ended up being a much simpler solution.

Is the Design Process Getting Easier?

I do think the design process has gotten easier throughout this course. I've gotten better at anticipating implementation problems and the methods required. Another thing I've learned is that specificity in sequence diagrams can go a long way. At work (PLC firmware development at Rockwell Automation), our sequence diagrams tend to be high level and "hand wavy", showing the interactions between components at a high level, but not concrete methods or objects. In this course, I've learned that creating sequence diagrams with real methods and instances is a great way to find problems in the design process, where the cost to fix them is much lower.

Design Review

Unfortunately in this instance of the design review, I didn't get much actionable feedback from my review partners, so this didn't end up improving my design.

I did offer some feedback to my partners. Some feedback was on a sequence diagram where the ordering of participants made the diagram difficult to follow. In this case, I suggested that reordering the participants in a roughly sequential order of when they are called would help the diagram be more usable.

I also offered some feedback on a class diagram for some missing relationships between classes.

Finally, I offered some feedback on some requirements I felt could benefit from more specificity.