Gerald Arocena

CSCI E-97, Fall 2019

Assignment 2

Document

Creating the design helped a lot with making the implementation easier. It helped me structure my classes in way that mirrored the class diagram. Visualizing the design document's diagrams in my mind made implementing my design more efficient. I became adept at pinpointing what class or area in my code corresponded to the visual diagrams and vice versa. This was the first time I coded a program with such structured modularity and it saved my time and effort I otherwise might have wasted. It also clarified what was needed in this Model Service module and what wasn't which also helped as I implemented my design. I kept in mind what later modules would need from the Store Model Service which hopefully will save me from undue time and effort later on.

My design could have been better in a few ways. In particular, I don't think I considered the CLI syntax commands enough before I made the design – something I realized late in my implementation but luckily wasn't a huge error. For instance, upon examining the CLI syntax for "show" commands further, I realized that I had scoped access to the Inventory, Sensor, and Appliance classes somewhat inefficiently so I changed my original associations to them from being just with the Store class to also with the Model class. I also changed the Basket association in my original design because I lacked the ability to see how to associate it correctly with the system before my implementation. For whatever reason, I could not conceptualize how the Basket class would fit. Originally, I made the seemingly logical choice to have the Customer class associate to the Basket class and then changed my design later on to having the Modeler class associate to the Basket class exclusively, since customers are globally scoped (I used customer id as the basket id). But I may have needed some time with implementing my design before I was able to understand that design choice clearly. None of these changes had any negative affect on meeting the Store Model Service requirements. The changes I made to how objects are scoped allowed for a more concise implementation solution. And concerning the basket change, the customer class is still associated to baskets through the use of customer id's as basket id's. In the Modeler, I created a mapping of "active baskets" from basket id (same as customer id) to a Basket object which worked out well.

The design review was also very helpful in improving my design. I was able to see what others had done that I had missed. Likewise, I pointed out requirements that the others had missed. This helped prevent

me from losing points on the assignment I otherwise might have lost. For instance, I confused the Overview section in the design template with the overview section mentioned in class and on the class slides which my peer reviewers pointed out to me in addition to my grammatical mistakes. I also got feedback on one of my “defineStore” method parameters which was also helpful advice.

Below are the comments on my design document made by Sean Bollin:

> Hi Gerald - most of the textual areas of your design doc look great to me. I think for the class diagram you'd probably get points deducated for not putting the properties and methods in the UML diagram itself. Also there was a request for another diagram of the entire system that I did not see in your design (like a diagram of how all the components interact - including the controller, authentication, etc.).
>
> For the createStore API - I'm not sure about accepting an Aisle and a Shelf as an argument. If you want to create a store and you don't have an aisle or shelf, then you'll have to pass "null" in which is not desirable. Also, what if you want to create a store with multiple aisles and shelves? Then you're left in an akward situation of creating a store with only one initially.

Below are the comments on my design document made by Nicholas Antwi:

> Hi Gerald,
>
> Thanks for your feedback, I looked through your design documents and it looks good to me, but there are some things that you have to change them, so when you get a chance then you look at them.
>
> 1) Future tense
> According to the lecture, we should use present tense instead of future tense. Initially, I wrote my in the future tense but I have to changed everything into present tense, after the lecture; and that was what I was doing last Sunday. So sentences such as,
>
> **Provisioning Stores:**
> 1. “1. The Model Service **should be able** to process provisioning commands for configuring a store’s layout, architecture, inventory, customers, and other assets.”
>
> Should probably be change to:

1. 1. The Model Service **is able to** process the provisioning commands for configuring a store's layout, architecture, inventory, customers, and other assets.

2) **Manage Modeler**

"Each store should be able to retain **it's** own identity and functioning without undue interference, e.g., from other stores under its service."

I think " **to retain it's own identity** " should be (**to retain its own identity**)

3) **Class diagram**

I think the methods, properties, and operations should be included in the UML diagram.

Thanks again,
Nicholas

Below are the comments I made on Sean Bollin's design document:

Thanks for the feedback, Sean! So for yours, it looks great but had a few things:

- Consider adding a sentence or two to the Introduction. In class he explained that we needed an "overview" (different from the Overview in the design document template that includes a diagram) that describes how the document is organized. In the grade sheet: "presents a short description of the organization of the document."
- You said, "Adding, removing, and clearing products from a basket must also update the corresponding inventory." My understanding was that inventory should be cleared/updated at checkout. Not sure if updating inventory with the basket would hinder your implementation or not for this assignment and later on.
- Assignment 1's design document mentioned explicit methods where exceptions were thrown. Maybe mention where exactly important places need exceptions per the requirements?

Overall, clever design, great following of the design template, makes great sense, and great quality! I'll let you know if I find anything else worth mentioning. Good luck! -Gerald

Below are the comments I made on Nicholas Antwi's design document:

Hey Nicholas,

Your design doc looks great, but just a couple things:

- Great Introduction and Overview sections, but consider adding a sentence or two to the Introduction. In class he explained that we needed an "overview" (different from the Overview in the design document template that includes a diagram) that describes how the document is organized. In the grade sheet: "presents a short description of the organization of the document."
- Possibly consider condensing your Requirements section. The design document template said it should be "brief and to the point"; a summary basically.

- The requirements also stated that a store must have at least one aisle and shelf. Not sure if you mentioned that.
-Consider including asterisks in your class diagram to make one-to-many relationships clearer like in Assignment 1's doc and as mentioned in lecture.
- Assignment 1's design document mentioned explicit methods where exceptions were thrown. Maybe mention where exactly important places need exceptions per the requirements?

Overall, very-detailed and thorough. Great effort, understanding, and job! -Gerald