Reflection
Assignment 1
Tyler Forrester

Design Document

Please refer to the design document for a complete description of design choices and class set up.

Reflection

The design document phase of the process worked very well setting up use cases for the program. This clarified for me how to approach building the program. Especially adding an item to an array and deleting that item. It was nice to imagine how my program would work before it was built. However, I didn't go far enough in my design. Instead of only addressing main uses cases like displaying a list and adding an item, I should have also spelled out my cases for user error. Use cases like "trying to delete an item, when they are no items" took equally long to code and should have spelled out specifically in the design document. I also think that defining the functions in the design document was good, but the description could have more brevity due to the fact I had a lot of redundant verbiage between the use cases, function headers, and class descriptions. Ideally typing the same information three times should be avoided, so that I have more time to working on the other projects due this week.
Formatting the data should have also been covered in the design document. I found myself spending to much time trying to format the program in the code. It would have been much easier to have had a plan in the design document and then been able to check my grammar and spelling against it.

Coding the Project

Reflection

Coding the project went well for the most part. My design document was thorough enough that the implementation went fairly easily. Formatting the data and testing for user error were the two things on which I spent the most time. I also made a mistake in my original design choosing to use and array of pointers that point to different item objects rather than a pointer to an array which holds different items. This was a time consuming mistake to debug, because the I could addItems to the array and I could delete Items from the array, but I wasn't able to delete my dynamic array in the standard manner and repopulate as the Professor and TAs described. I assumed my mistake was in the doubleArray function and spent four hours trying different strategies to delete then recreate the array without a segfault. On future projects, I intend to set a time limit and then look outside the function for errors. It was a time expensive task trying to fix what was working perfectly fine.

Testing the Project

Overview

The testing of the input revolved around the idea of a user using my program. This user would try to stay within the bounds of the program, but if he got lost or confused or fat fingered, I needed to make sure those outcomes were handled as well. My testing took every use case that I could find and tested it. From user's choices being too high or too low to user selecting an item such as delete item that wasn't enabled or user selecting a duplicate name. These cases cover the program to the best of my ability.

Please see testing document for tests.

Reflection

Testing the good input was easy since I had already written up use cases for the program. Testing bad input was much more difficult because I specifically spelled out how bad user input would be handled. I instead had written a class to validate the data without adding it to my design document. This made testing the "bad" use cases difficult. Then I discovered if there were no items to delete that the user was unable to exit the delete array function and I didn't have a test case for it! This forced to me back into the design and coding part of my project to account for this sort of use case. Once I had added some tests around nonexistence arrays I felt confident that my program was completed so I exported my files and zipped them.