Data Structure Design

Dynamic Arrays

Dynamic Arrays will be used in this inventory system to store many “Item” Objects, which represent merchandise. This data structure was chosen because items should be accessed and edited more often than written to or removed from. This makes the dynamic array a better fit for the object because they are able to access arbitrary items more quickly than linked lists. We will not need to implement a method to edit an array item, because setter can be accessed from the array. This data structure could also be replaced with a map/dictionary.

Tree Structure

Trees will be used in order to track categories and subcategories of items. This will be a general tree structure. This is because categories will be created by the user when they create or edit an item. This will be done by typing the category in text, broken apart by a special character. This special character will likely be a “\” to mimic file structure notation.

Object Design

Items

Items will represent merchandise within the inventory system. This object should primarily be used to store values. These values include ID, name, quantity, price, and category. Functions for this object should include getters and setters.

User Interaction

Console

The base implementation of this project should be done through the console. This is for simplicity in development and testing. The user will interact with the console through single key instructions given by the program The actions the user should have access to are as follows:

- [l] List Items:
    - [a] All Items:
        - Response: list all items in the array with their associated values.
    - [c] List Categories:
        - Response: Print category tree, using indentation to represent children of nodes. Use item names to represent leaves
- [s] Search Items:
    - [n] Name Search:
        - Input: A string to be compared against names of array items.

        - Response: print a list of items and their values that may match the user's input.
    - [c] Category Search
        - Input: User will input the category they want to search, starting from the first node after the root.
        - Response: Print a list of all children of requested node.
- [c] Chose Item:
    - Input: Requires the number corresponding to the ID of the desired item.
    - [e] Edit item:
        - Response: Allows user to edit item, there are two acceptable implementations for this.
            - Use a library that allows the console to be prefilled for input (such as prompt-toolkit)..
            - Show the user current value, allow the user to press enter to skip or type to change the value.
    - [d] Delete Item
        - Response: delete the item.
- [e] Exit:
    - Response: Save data and close program
- Enter
    - At any point the user may press enter with no input to go back (unless they are currently editing an item).

Function

Item Creation

All function should be handled within the driver. For item creation, the driver should create an “item” object and pass in the associated values. The item should be initiated from within the dynamic array.

Saving

This project will implement a saving system. Saving will occur at program close. This also means that when the program initiates it should check for existing data and prepopulate array with item objects. This should be done with csv, database format, or another similar format. When the program saves, it should delete the previous document.