CS240 Final Project: QuickShoppr

Prepared by: Kyle Peeler

April 5, 2016

PROJECT SUMMARY

Description of Project

QuickShoppr is a simple grocery list application that allows the user to organize a grocery list that will then be optimized to reduce the amount of time spent shopping inside a grocery store. The user will be able to save and load a grocery list, and each grocery list will contain grocery items. When a user adds a grocery item, they will be asked to add a tag to the item representing the category that item is within (for example, Apple is a Fruit). The user will be able to select which categories correspond with which aisles. Once this assignment of categories to aisles is complete, the app will be able to list the added grocery items in ascending order of aisles, allowing the user to only have to stop at each aisle once.

Intended User

The intended user(s) for this is to be my family. A big part of why I chose the field computer science is to make applications that help other people. In our family, we constantly have multiple grocery lists that sometimes get misplaced and thus we forget stuff at the store! This application would not only prevent that from happening, but also hopefully reduce overall time spent in the store by ordering/categorizing the items so we can get in and out of the store faster.

The Problem to be Solved

Shopping in a grocery store often requires you to back track through multiple aisles, simply because your grocery list is written in the order you decided you need the item. This application will store your grocery list and order the items based on the category the item is within and by the layout of the store (showing all categories in aisle 1, then aisle 2, then aisle 3, etc...).

Technologies Used

For this project, there will be several technologies that will be used. Each grocery list will be serialized as a file, allowing the user and Save/Load a grocery list. I plan on making a graphical user interface, as this application needs to be user friendly for my family members to use it. Finally, I plan on attempting to write this program using the Swift programming language, that way the application can be used on either an iOS device (the primary phone operating system my family uses) or on Mac OS X.

DATA DESIGN

What data is your program really about?

The QuickShoppr app is simply a list of Grocery Items. Each grocery item can be marked as completed, assigned a category, and given a name. Each list of grocery items will be grouped by their category. Before the list is displayed, the user will assign a grocery list to a grocery store. This grocery store will define what categories belong to what aisles.

What is the best way to represent the data?

The GroceryList will be a class that contains an array of GroceryItem elements, as well as the associated GroveryStore. Each GroceryItem will be an instance of the GroceryItem class with properties for the item's name, item's completion status, and item's category.

A GroceryStore will be a two dimensional array, with each index of the row dimension representing an aisle, and the column dimension representing the categories that are associated with that aisle. We will represent the aisle by the index of the row dimension.

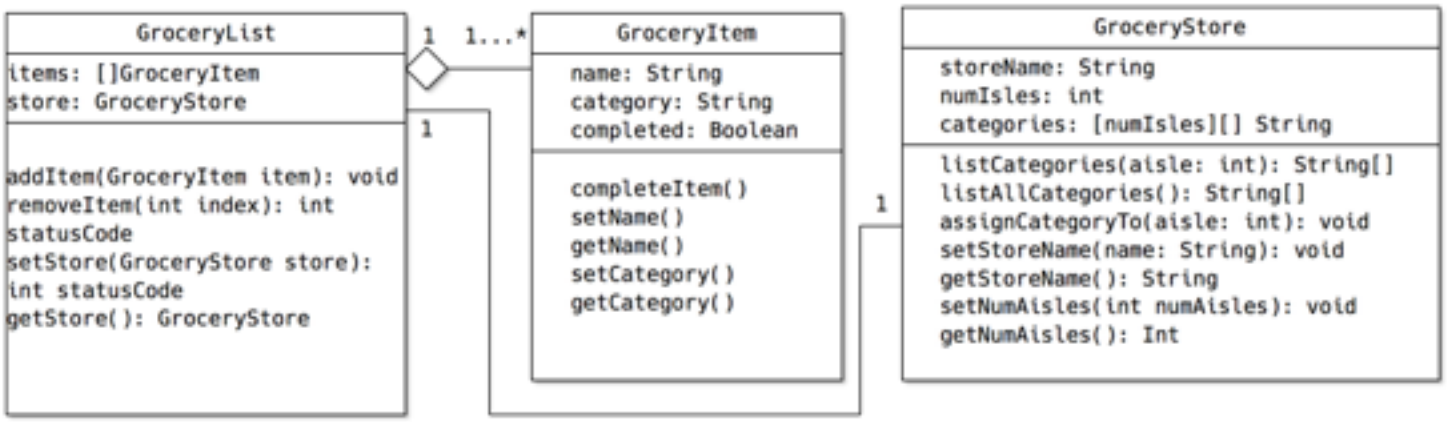
Will the data need to be persistent? How will you make that happen?

The GroceryList will be serializable so it will not need to be persistent within the application itself. I may design the application to automatically load the last opened list upon launch. However, the GroceryStores should be persistent with the application, that way a user will only need to declare what is within each aisle only once.

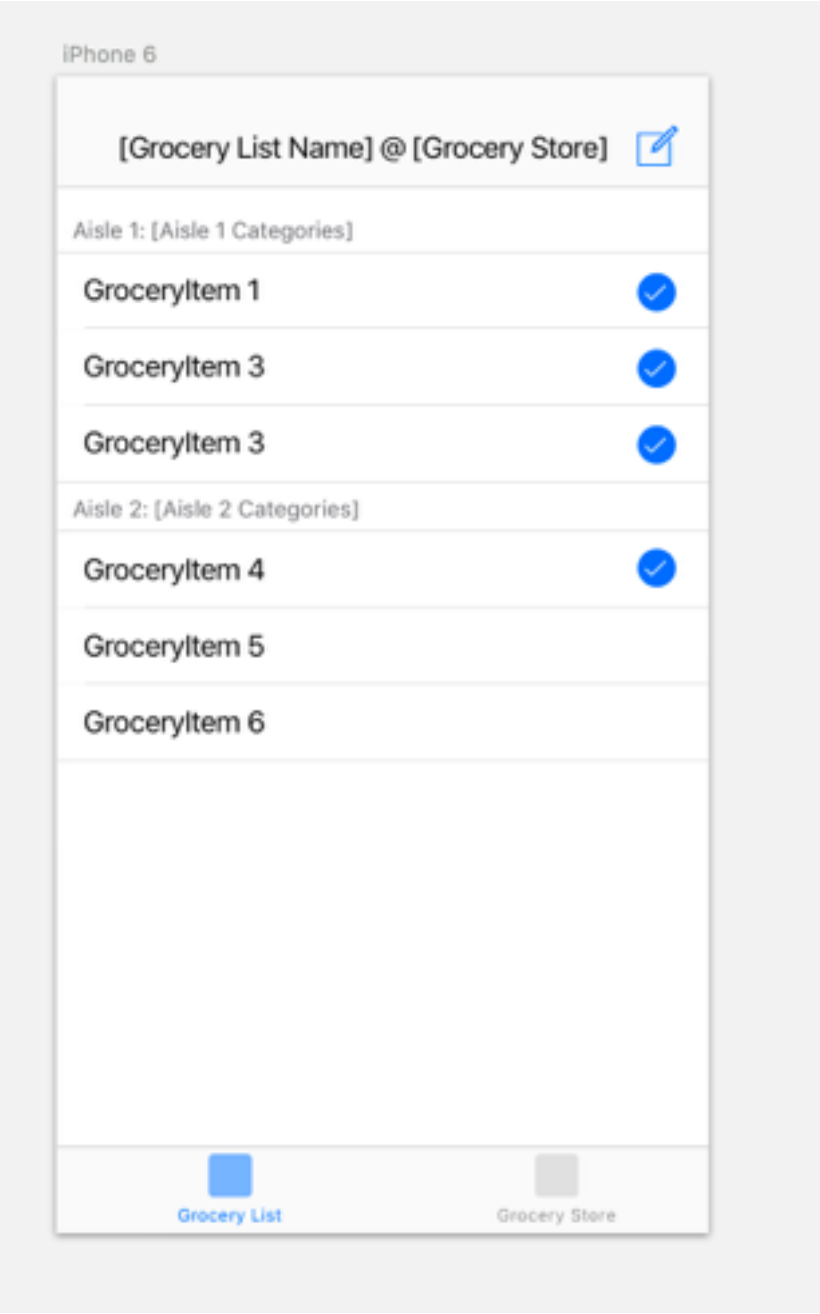
Will the data need to be aggregated into a larger structure? How

The two main structures that will encapsulate all of the data in the program will be a GroceryList which contains many GroceryItem elements, and a GroceryStore which will describe what item categories are within each aisle.

UML DIAGRAM


GroceryList
items: []GroceryItem
store: GroceryStore
addItem(GroceryItem item): void
removeItem(int index): int
statusCode
setStore(GroceryStore store):
int statusCode
getStore(): GroceryStore
1
1...*
1
GroceryItem
name: String
category: String
completed: Boolean
completeItem()
setName()
getName()
setCategory()
getCategory()
1
GroceryStore
storeName: String
numIsles: int
categories: [numIsles][] String
listCategories(aisle: int): String[]
listAllCategories(): String[]
assignCategoryTo(aisle: int): void
setStoreName(name: String): void
getStoreName(): String
setNumAisles(int numAisles): void
getNumAisles(): Int

UI DESIGN


iPhone 6
[Grocery List Name] @ [Grocery Store]
Aisle 1: [Aisle 1 Categories]
GroceryItem 1
GroceryItem 3
GroceryItem 3
Aisle 2: [Aisle 2 Categories]
GroceryItem 4
GroceryItem 5
GroceryItem 6
Grocery List
Grocery Store

ALGORITHM

What are the data members?

The main data member will be a list of GroceryItems contained within a GroceryList. Upon creation of a Grocery Item, the user will input the name and the category of the item.

The GroceryStore data member will define what item categories belong to what aisles.

Describe the initializer - Initializers always create and populate the data members. Will you read in parameters? Have default values? both?

The initializer for the GroceryItem will create a new grocery item with an assigned name and category, both inputted by the user at time of creation (thus both of these will be a parameter of the constructor). If a category is not inputted, the category of Uncategorized will be assigned.

On the creation of a store, the number of aisles will be required upon creation (this will be a parameter of the constructor).

Describe any other housekeeping that may need to happen in the initializer

None? (Not sure, will update)

Define any properties or virtual properties you class may need

See UML diagram

Identify any methods your class will need beyond access modifiers

See UML diagram