Aidan Duffy

MET CS 665

Boston University

Assignment 1, Task 1: Description

Flexibility

In terms of flexibility, it is my goal to ensure that this program is as flexible as possible. Therefore, there will be an interface for drinks as well as "condiments" so that it is easy to add or remove new types of drinks, like soda or juice as well as ice or artificial sweeteners. These new additions would simply be realizations that implement the drink interface.

Simplicity and Understandability

I wanted to ensure that the code was modular, utilized encapsulation, and properly marked, well named, and commented so that it was not cluttered, easy for developers to read, and simple enough to understand.

Duplication Avoidance

I wanted to use a condiments and drink interface so that there was limited duplication and that any unique aspects for coffee vs tea or milk vs sugar would simply be limited to their individual realizations of their classes.

Design Patterns

I utilized realization while using a drink interface, and the machine itself will be "composed" of all the possible options of drinks and aggregated of all the "condiments". This is composition because without any drink options, the machine has no function, but it can function without those condiments. As I mentioned before, I used modularization and encapsulation so that it was easily readable and understandable.

General Overview

I will have several packages: beverage(for the drink and condiment interfaces and all implementations) and machine (for the DrinkMachine class). The interfaces and realizations will be implemented as described above with associated prices, max quantities for condiments, etc. The machine will run the entire program such as asking the user for their drink preference, drink type, and condiment choices then dispense the drink. All of these will have associated JUnit tests.