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 abstract class 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 added through inheritance that are subclasses of the drink abstract class.

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. The abstract class and all the inheritance should separate everything in such a way that it will be easy to navigate and understand everything's purpose.

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 inheritances of their classes.

Design Patterns

I utilized inheritance while using a drink abstract class, 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 abstract class and all implementations) and machine (for the DrinkMachine class). The abstract classes and implementations 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.

Running the Program

Run the relevant maven commands on machine.MainAssignment, not the Main.java file.

Note: As I am about to wrap up my project, I am realizing that I vastly overcomplicated my design from the get-go, and had I planned further ahead (was away from my home and computer until Sunday), I would've had more implementations to test with. So, I apologize for any confusion, as well as some of the bugs mentioned by the maven scan, though those don't seem to impact functionality. I will be quickly starting on the next assignment to prevent this from occurring again.