Aidan Duffy

MET CS 665

Boston University

Assignment 2, 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 separated classes for the stores, the requests, and the drivers. I will ensure that these are generalized such that any store can be added as well as any new drivers. There will be lists of these objects so that new objects are easily added and any store front can create a new request that is sent to all drivers.

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. All of the classes will be generalized enough so that new objects can easily be created.

Duplication Avoidance

Given that all of the major objects will have entirely separate classes, and they will have little to no overlap, then there will be little to no opportunity for duplications.

Design Patterns

I utilized the observer design pattern as that made the most logical sense. When a store creates a new driving request, then all of the drivers will be notified of the state change as well as when a driver accepts/is assigned to take this new request. In that sense, the delivery request as the subject and all of the drivers will be the observers.

General Overview

Currently, the program operates from a simple observer design. A set of shops exist as well as a set of drivers. A single, central DeliveryRequest exists, and it is either null/inactive or it is open and searching for a driver to be assigned to it. Once it is assigned to a driver, the request object still exists, but the central subject request will return to a null state until a shop creates a new request order. The system exists such that one request is processed and assigned to drivers at a time, in a first come, first server order. Eventually, it would be nice to integrate some metric, like location as well as threading so that it could operate more efficiently. In its current state, once a shop creates a request, this central request object notifies all of the drivers, assigns itself to the first available driver, then once the driver accepts this request, the central object returns to null, once again notifying all drivers that are available and searching for requests.

Running the Program

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