About

> Written by Liam Major, 30223023.

This is a (more or less) full explanation of how system that I spear-headed works.

> I took the idea from Andre Beaulieu, thanks Andre.

I'd also recommend instead of looking at the PDF that was included, view the document online in LucidChart for better viewing.

I've included a git log as a text-file and a full list of people, UCIDs and Github usernames in a file called `Authors.txt`.

Task Delegation

- team leads
  - Liam (development and overall project oversight)
  - Sinan (development)
  - Ali (QA)
  - Sajid (documentation)
- development team
  - Alecxia
  - Gurjit (GUI)
  - Kelvin
  - Liam (team lead) (GUI)
  - Logan (GUI)
  - Nezla (GUI)
  - Ohiomah
  - Sinan (team lead)
- QA team
  - Abdullah
  - Ali (team lead)
  - Emmanuel

    - Falah
    - Michael
    - Umer
- documentation team (UML)
    - Adefikayo
    - Maleeha
    - Anmol
    - Ana Laura
    - Sajid (team lead)
- demonstration team
    - Liam
    - Gurjit
    - Emmanuel
    - Logan

Running the Program

Part of the development for the GUI was done using IntelliJ, I've included the necessary `.class` files in `com.thelocalmarketplace.software/lib/classes/`, for whatever reason, this isn't automatically added to the class path of the project.

In order to run this project, you **need to add these files into the class path of the project**.

System Explanation

There are various elements at play here:

- managers
- observers
- utilities

The system is setup as such in order to give some sort of structure to the overall view. In my mind, it makes sense to create these (seemingly) arbitrary rules and restrictions in order to figure out what to do, what not to do and where to do certain things.

I heavily operate on the Unix philosophy e.g., a certain thing should be really good at doing that one thing and nothing else. If, for instance, I need something to do multiple things, I can break this apart into various other smaller functions and use those smaller functions to do something bigger.

Managers

What is a manager? They are essentially facades into the lower-level parts of the hardware. The goal of a manager is to do one task effectively:

- `OrderManager` manages items
- `PaymentManager` manages payment
- `AttendantManager` manages the attendant and status of the dispensers and other components
- `SystemManager` owns all the other managers and decides when and when not to allow certain actions to happen

Every manager has it's own interface, this tells everything else in the system what the manger can do and what it cannot do.

The `SystemManager` delegates almost all of it's functionality to it's child managers. It has all the functionality of every manager because it implements every manager interface, including its own.

Interaction With the GUI

The gui related objects are purely listening and interfacing with the system manager and nothing else. They really don't decide anything, rather, they just tell the system manager to do something and, in response to the system manager, perform an action.

All gui related objects receive notifications from the system manager from the `ISystemManagerNotify` interface.

Observers

The observers simply monitor a given object and notify its manager as to what happened and the state of the object that's being observed. For example, if a printer runs low on ink, the observer will receive the event and notify it's manager.

From there, the manager decides what to do with it. The observers do nothing but tell the manager what happened and give it any and all relevant information.

The observers notify the manager by their `IXXXNotify` interface.

Utilities

This refers to anything in the `com.thelocalmarketplace.software/utils/` package. These create the things that are required for the system to function.

For example, if some part of the system needs a `Card` object, it will use the `CardHelper` class. If the customer wants to add an item (because of constraints), the `DatabaseHelper` will randomly create a new item and product pair and store that in the public facing database.