SENG 300

Group Development Assignment 2

Group 4

Design Justification

The main goals of our implementation were to provide a solution that allowed for simple testing, as well as flexibility to change in requirements or the vending machine hardware. The agreed upon overall solution was to take advantage of the fact that the vending machine configuration closely follows the observer design pattern since the subject-observer relationship between the vending machine and its components was already established by hardware. The benefit of going with the observer design pattern is that changes to one component of the hardware do not have a large effect on the other components, and the relationships between all the different vending machine parts became much simpler to deal with.

Another point of emphasis for our code was modularization, for simplicity and for code re-use. The benefits of having all our methods be so short and simple can be seen in a few places. The separation of toggle methods for all the lights avoids a-lot of potential redundant code repetition, because there are multiple places in the code where these methods are called and without them our code would contain lots of redundancy. Another advantage is in terms adaptability to change. If there is a change, be it in hardware or the requirement, had we written large complex methods odds are we would have had to trash most of them; whereas, with small modular methods there is a greater chance more of our code is still usable.

Flexibility of the code to support credit card payments was also something we considered in our design. The credit is accessible globally so that we can add a method to increment it via credit card when required. This allows for mixed modes of payment which will eventually become a requirement, and it doesn’t require a change in the logic for how we vend our pop since the methods for calculating change are all separated.

In terms of changes to the communication with the user, we took the same approach to separate all the processes that involve communication with the user such as the timer for the display, and the toggles for all the lights. If there were a change in the message, we want to display it wouldn’t take much change to the code. In respect to the event log changes the user wants to make are also just as simple. If the user wants multiple log files, a limit on the size of the file, or even a different file type for the log, all those changes can be made super easily through arguments in the file handler constructor.

The way our constructor is defined taking in an instance of a vending machine as a parameter allows for greater flexibility, because it will work on a variety of vending machine configurations without having to change anything. Since none of the values for coin-rack size or any of the other properties of the vending machine are hard-coded our code is adaptive to the vending machine that is passed into it through the constructor. This avoids us having to change anything for vending machines of the same configuration with different presents.