**Design Justification for Group Assignment 4**

In regards to functionality, the new updates administered to the vending machine now supports alternative payments(such as the ability to pay through debit card), a graphical user interface, and a configuration panel for technicians to perform certain tasks such as changing the price of pop, or load/unload coins/pop from/to the vending machine. It also includes a locking and unlocking mechanism, which is designed to enhance safety protocols for technicians, as the configuration panel can only be accessed when the machine is locked.

Our updates are implemented through a number of new classes that communicate with each other; specifically, they communicate with their respective logic/listener classes. For example, the configuration panel GUI communicates with the newly added *ConfigPanelLogic* class - where most of the logic for the configuration panel is handled. And the main GUI uses the pre-existing *VendingMachineLogic* class from the previous assignment to incorporate its functionalities. By utilizing the pre-existing logic software as well as having a semi-closed system, our software is made to be less dependent on the hardware side of things. This lessened dependability between the software and hardware increases changeability and adaptability of our program, while providing increased comprehensibility in case changes do need to be made. This takes the advice that was provided by Mr. Client on keeping the software at optimal level of changeability without the burden of the aforementioned hardware.

Furthermore, our organization of the source code is formulated in a manner that can be easily understood by an individual with a Computer Science background. Classes are created in groups according to their tasks/functionality, for example, the source code has classes that all fall under the graphic interface, the configuration panel, and so forth. This straight-forwardness and closed design makes the logic of the software easy to understand and the software independent from the hardware. This allows for accessibility of new implementations in future updates. This manner of programing through semi - closed systems would prove convenient, as it would not interfere with any recent changes to the software. When communicating with one another, via vending machine’s software and hardware, the two is akin to the likes of a TV communicating with a DVD. If in a case that an aspect of the DVD is flawed, the TV would not experience negative feedbacks from major bugs, but instead maintains its integrity by remaining as independent entities - despite the awareness of the other system’s presence.

In addressing changeability, the most realistic feature allowing this code to evolve would be the ways in which the user’s credit is incremented. Currently we are keeping track of the user credit in a variable, where the credit is incremented when a valid coin is inserted, or when a user pays with an alternate form through calling the method *addCredit()* in *VendingMachineLogic*. With this method, where one universal variable keeps track of the credit, we can add features that allow a user to partially pay coins, and pay the remainder with any other form of payment, including via debit card. This method keeps the flexibility and evolve-ability of the code very high, thus making it easy to implement users paying with various forms of payment, including Bitcoin. Once again, the idea of closed systems applies here as the calculation of the user’s credit is done through inside the *VendingMachineLogic* class. This as a result will not affect whether or not the user is inserting a coin, or paying with a debit card, or some combination of each.

The efficiency of the machine, in respective of a company’s perspective, is connected to its adaptability and changeability. With this idea of efficiency in mind, we have added in a configuration panel where the technician can perform certain maintenance tasks (i.e., change the amount of money required for a pop-can, lock/unlock the machine, load/unload coins or pops). With safety as a priority, we made sure that the configuration panel can only be accessed while the machine is locked –enabling the safety and disabling the hardware components of the vending machine.

As an entire entity, the structure of the vending machine’s software is comprised of interconnected closed systems, thus decreasing dependencies and allowing adaptability to new hardware. By having the ability to support a new hardware with some minor adjustments, and through its extensibility to function with multiple forms of payment, the vending machine’s software has a potential to scale to future technologies. The groups of closed systems not only allow for high levels of changeability, it also has been proven through the examples we discussed above. In confirming this structure’s validity through levels of immense changeability, the simplicity in understanding the software is also observed. With heavy documentation to support one’s understanding, the code does not require a convoluted amount of tracing to be understood. Instead, one will only be required to look at each closed system, and understand the modes of communication occurring throughout the machine. Conclusively, this design in itself will enable the machine to be easily updated, and bring an advance towards future innovate ideas for vending machines.