# SENG 301 L01 - (Winter 2017) - Software Analysis and Design A4: Design

You've been on vacation for a few weeks after scrambling to get the unit tests together. When you get back, your boss has just gotten back from CES, where she has a brand new idea: the vending machine should be able to sell way more than just pops -- it should be able to dispense food items, or tissue paper, or whatever else! In addition, it should work with all sorts of different kinds of payment (e.g. loyalty cards, credit cards, tap to pay phone stuff, etc.), and in addition to that, it should work with new kinds of display devices (e.g. augmented reality glasses, cell phones, etc.). What!? Actually, your boss hasn't figured out which of these things she wants the vending machine to work with yet.

So, your task is to redesign the system such that *when* the decision is made, it will be easy to "plug-in" any one of these new payment, communication or hardware in the future. This means that adding a new module means adding a new class, and not re-writing the other parts of the system.

Your job with your (up to) three-person coding team is to redesign the system (by adding classes):

1. Where there are new façades that simplify interaction with things like Funds, Communication, and Products.
2. Where the business rules around "what happens when someone selects a product" are wrapped independently from the hardware.

This is focused on the following three design goals for extensibility:

- easy extensibility to other forms of payment, including mixed modes (e.g., paying partially with coins, partially with credit card)
- ease of changing hardware for communication
- ease of supporting alternative hardware (this is already done)

## Learning Objectives

By the end of this assignment, you will:

- Re-design a system to make use of the façade design pattern
- Use event handling, and the observer design pattern to communicate between modules (ha!)

## Deliverable / Submission

You will deliver a .zip file into the Dropbox on D2L containing:

- source code that supports the above requirements and design goals;
- a test suite that tests the functionality (this can just be the existing test suite, or you can augment it, but the coverage should be the same or greater),
- a 1-page justification of your design, relative to the design goals above.

# Steps

1. Download the A4 skeleton
2. Read through the documentation below
3. Without modifying any of the existing code from the Vending Machine project (aside from VendingMachine.cs), add additional classes to implement the facades
4. Submit via D2L (indicate who is on your team). Only one submission per team.

# Description

The focus of this assignment is on design. Here are some slides that discuss this assignment. The purpose is to think about how to make a design more extensible once you already have some working code. This is surprisingly uncommon -- i.e. that additional functionality is requested later in the project, or that you discover you can make the code more general to work with scenarios you hadn't anticipated at the outset.
One of the ways to handle this new kind of complexity is to consider using the Façade design pattern, which allows you to simplify interaction with a set of underlying classes through a more simple interface. These Façades can manage interaction with other parts of the system through events--subscribing to them allows different components to "listen in" on when interesting things are happening, triggering new interactions (e.g. function calls) with other components.
You'll notice that in Frontend4.Hardware, there is a new class called HardwareFacade -- this is the facade over top of the hardware as a whole. You'll notice that most interactions with the hardware can be done essentially with this class.
Your redesign really needs to focus on:

1. a façade for handling interactions with payment (e.g. to allow for new forms of payment -- such as credit cards, debit cards, etc.), where in principle, it handles partial payment from each of these sources of payment;

Note: do not actually implement any of these other forms of payment -- just make it easy to do so if one wanted to (i.e. by creating a new class without having to re-write the system).

1. a façade for handling interactions with the products -- i.e. dispensing the products or refilling the products in the vending machine;
2. a façade for communicating with the user -- i.e. "displaying" what has been selected or vended, and
3. a class for handling business logic (e.g. what the costs are for different products; how excess payments are handled; dispensing the product; communicating with the customer, etc.) -- this is one that perhaps ties the other facades together.

Thinking carefully about what each of these different components is responsible for will give some clues about what type of functionality and communications are required of each.
You may assemble a team of **up to three people** (this means you can work independently, or with a partner, or with two partners). Ensure that you all share the same TA.

## What's Changed

Three "major" things have changed in Frontend4 from Frontend2:

1. Cents has been added as a new abstraction on top of money. Before, we worried only about Coins, and thought about value in terms of coins and what value the coin represented. To allow for "payment" abstractions that don't deal in "Coins" specifically (and instead in "Cents"), we needed a new way of representing value--hence, Cents. Coins are now built with Cents as the underlying representation for value.
2. HardwareFaçade is a new class that used to be the old VendingMachine. We now consider VendingMachine to be an even higher-level abstraction above the hardware--something that manages business rules around interaction, products, what is communicated with users, and so forth.
3. Product instead of PopCan. Whereas before, we were primarily concerned with pop, we are now able to vend products in general.

You'll note the test methods have already been built on your behalf. These ultimately should work once your design is complete.

## Questions

Post your questions to the #A4 channel on Slack.

## Grading Rubric

75% of your grade will be based on your new design (as implied by you implementation and justified by your written explanation). 25% of your grade will be based on testing.

## No Late Assignments

Your assignment is due at noon on the specified due date. No late assignments will be accepted.