This document details my experiences with this Vending Machine project. Everything from problems encountered and solutions employed, to design choices and lessons learned.

Problems Encountered

Problem: Using an enum for Currency, and C#'s inability to use non-integral types (such as decimal) in enums.

Solution: Simply switched to representing my currency values in whole cents instead of fractional dollars. No need to worry about rounding errors, and can easily be converted to fractional dollars if needed in the Vending Machine message outputs. There are a number of different approaches I could have used in order to allow the usage of decimal representations of Currency values, but I wanted the experience of working with enums in C#.

Problem: Accepting a transaction regarding being able to make change after the fact: (example: Till holds 3 quarters. I insert one dollar and purchase something worth 70cents. It's impossible to return 30cents)

Solution: I implemented a method to check if change can currently be made for a given value. This is used by the vending machine to determine if change can be made after a purchase before the transaction occurs. (example: going back to the problem example, the vending machine simply disallows the selection of the 70 cent item.)

Notes

On my experience with Test Driven Development: I tried to employ TDD practices where I could, given my inexperience with the concept. I feel like this went quite well with my Till class. I was able to think through method stubs that I thought a Till class would need. Then I wrote tests for these methods. Then I implemented the method bodies. The TDD approach reduced the number of bugs encountered and reduced the difficulty of fixing bugs if one was encountered. I only had to set breakpoints and step through my code a handful of times during this entire project, and a fair portion of those events were during the creation of the UI portion which isn’t covered in tests – so I’m quite pleased with how a TDD approach improves the work.

On design: So, this feels quite embarrassing, but I am pretty sure I designed this project backwards. One of the first things I did was ask myself “What does a Till / Inventory (component classes to VendingMachine) need to do?”, when I should have first asked “What does a Vending Machine need to do?” and then built off of that by dividing the functionality into things that component pieces do. Since I took the approach I did, I ended up spending time creating a lot of methods that wouldn’t be used. In whatever my next project ends up being, I intend to try starting with the skeleton of the final piece and then design the components to cover only what the final piece needs.