Capstone 1: Vending Machine

Team 8: Chelsea & Chris

Category	Feature	Score	Notes
Features	Requirements	3	• All requirements were met! Great job!
	Program startup VM Creation and load Sharing of VM	2	• Program.Main creates a MainMenu and runs it. Menu creates a VM. In the VM constructor, VM uses VendingItemManager.GetVendingItems to load itself. GetVendingItems has a hard-coded file name to load into a dictionary. • I generally like this approach, but VendingItemManager should not have the file hard-coded. It should be passed in, preferably all the way from Main. That way, multiple vending machines can be created which have different items in each one.
	How change is made	2	• Money object calculates change, that is good. • Money.GiveCHange returns a string which is ultimately shown to the user. This is better than doing a CW, but would be better if could simply return the number of quarters, dimes and nickels, and let the menu class form the message to the user from those values.
	How is product selected and dispensed	3	• VM.ItemExists and VM.GetItem are used to check for and purchase an item. These are called from the PurchaseMenu. This works nicely. The only thing I don't like about it is that the menu then infers what error occurred if something goes wrong. For example "if the item exists, and the money provided is enough to buy it, but GetItem returned false, then the item *must* be sold out. This seems a little fragile. I think it would be better if GetItem either returned a code indicating the error, or throws a specific exception.
Architecture	Use of OO techniques	2	• Mostly UI is encapsulated in the menus, and are not in the Item and VM classes. Except for VM.DisplayItems, which uses CW. • Gum, Chip, etc. derive from VendingItem. This is nice. The way it is written, there are two public properties on these items which mean the same thing: "Message" and "MessageWhenDelivered". Message should probably be private. • Encapsulation: on VM, readonly Money is nice. The various message strings (NotEnoughMoney, etc.) should be private. On VendingItem, all properties are public get/set and probably should not be. Money class did a better job of get/private set.
	Error Handling		• There are no custom errors to handle. File operations and such use try/catch properly.

Maintainability	Code comments	0	• Almost ZERO comments in the code.
	Testability of code	2	• Because there are no Console Writelines in VM and other classes you can test these well. However, they often return strings, so your tests are looking for specific strings: “Sorry, please insert more money into the machine to complete the transaction.” It would be better to check for a result code or exception condition. If later you decide there should not be a comma in the above example, the tests break.
	Tests	3	• Nice selection of tests.