Capstone 1: Vending Machine

Team 4: Amy & Karina

Category	Feature	Score	Notes
Features	Requirements	3	• All requirements were met! Great job!
	Program startup VM Creation and load Sharing of VM	3	• Love your Out of Order message! • A Stocker is created to load the file. A Vending machine is created, and then the stocker is used to fill the slots. Then the menu is called, passing in a reference to the VM. Good separation and will allow for flexible loading and testing. • You are recursively creating a new Main menu from the PurchaseMenu method. This is not good as the stack is filling up. When the user exits, he must press an additional exit for each time that the purchase menu was called during the program.
	How change is made	3	• There is a MoneyChanger class that does the work. Nice. • VM.MakeChange doesn't really make change, just sets the balance to zero. Would be a little better if MakeChange returns a MoneyChanger to the calling menu.
	How is product selected and dispensed	3	• Vending Machine does most of the work. Great.
Architecture	Use of OO techniques	3	• Really nice separation of duties for your classes • Balance should be encapsulated instead of public set.
	Error Handling		• No error handling, other than the Out of Order message on load.
Maintainability	Code comments	1	• You have a single comment for most classes, but nothing to document the code inside.
	Testability of code	3	• Well separated logic, should be very testable. • You probably cannot test bad selections, because all that code is in the menus.
	Tests	2	• I like that FillSlots uses an in-memory stock list instead of the file. • DispenseItems can't be thoroughly tested (error cases).