Vending Machine

- Create a CRUD application using MVC and Spring for DI - enables loose coupling
- Must use IO to store and retrieve data
- Take in user funds - return selected item and appropriate change broken into coin denominations
- Have full set of unit tests for the DAO and Service Layer
- Apply exceptions to allow for “graceful failure”

The View

- The MCV later that is responsible for interacting with the user.
- Has dependency injection of the UserIO
- Will display a welcome message to the user.
- Will display a dynamic menu of all Items marshalled in the item.txt file including the slotId, name, and price of the Items. If an item's quantity is 0 it will display that it is out of stock.
- Takes in a double for the user funds and converts it to a BigDecimal to be passed into the Controller. If the user inputs a number <= 0, will display an error message to have them input funds again. If user inputs am invalid character the View will display an error message and prompt user to re enter funds.
- Will prompt user to enter Item selection based on the slotId. Will take in value as an Integer and will convert it to a String to be passed to the Controller. If user inputs an invalid character it will prompt them to re enter. If they enter an invalid Item the machine will display an error and return the user's funds.
- Upon completion of purchase View will display the Item purchased and will return the user's change in the appropriate denominations and the exit message.

The Controller

- The “brain” of the application - tells the other layers what to do
- Has dependency injection of the View and the Service Layer
- Contains the main run( ) method
- Tells the view to display the welcome message, the items, and the, menu
- Will continue to run while keepGoing = true (while the user choice isn’t exit)
- When the user chooses to insert funds and make a selection:
- Takes in the user funds and their Item selection
- Tells the service layer to getAnItem based on the slotId received from the view
- Passes the Item and the user entered funds to the service layer in the PurchaseItem method and receives a ChangePurse in return
- It then tells the View to display to the user their Item, change, and the program exit message
- Controller will also catch exceptions and if it does will tell the View to display the appropriate message, return the user’s money, and display the exit message

The Service Layer

- Is an implementation of the Service interface.
- Is the middle man between the Controller and the DAO.
- Has dependency injection of the DAO.
- Has 4 methods that are called by the Controller:
- loadMachine method - tells the DAO to load all Items marshalled in the txt file.
- getAllItemsInMachine - tells DAO to get all Items - returns a List of those Items.
- getOneItem - takes in a String of slotId - tells DAO to get the corresponding Item - returns that Item.
- purchaseItem - takes in a String of slotId and a BigDecimal of money as its parameters. Gets the Item that matches the slotId from the DAO. Checks that the Item exists, that the Item is in stock, and that the user has enough money to make the purchase - if not it throws the appropriate exception. Subtracts the cost of the Item to calculate the total amount of change. Creates a new ChangePurse and passes in the amount due back to the user as its parameter. Subtracts 1 from the quantity of the purchased Item. Tells the DAO to saveAllChanges. Returns the ChangePurse to the Controller.

DTOs

- This application has two Data Transfer Objects:
- The Item:
- What will be sold in the vending machine. Has properties of: String slotId, String name, BigDecimal cost, and int quantity.
- Will be marshalled into persistent storage.
- The ChangePurse:
- Contains method that can be called to to calculate the amount of Quarters, Dimes, Nickels, and Pennies that will be returned to the user.

DAO Impl

- Is an implementation of the DAO interface
- Handles the loading and saving of the Item objects - accomplished by marshalling and unmarshalling.
- Passes the persisted Item data into the service layer.
- Handles adding, getting one Item, getting all Items (List), updating, and removing an Item.
- Mashalling - Item objects will be stored as text - split each property by concatenating with a delimiter
- Unmarshalling - Strings of Item data are converted back into Item objects - broken apart at the delimiter - values placed into their appropriate properties using tokens

DAO Stub Impl

- Implements the DAO
- Supplies canned data that can be used for testing.
- Allows testing of the business rules and the integration between the Service Layer and the DAO.