Assessment: Vending Machine

Re-submit Assignment

Due May 14, 2018 by 9am **Points** 100 **Submitting** a website url
Available after May 10, 2018 at 5pm

Overview

Your assignment for this lab is to create a program that simulates a vending machine. Your program must have the following features:

Features

- The program should display all of the items and their respective prices when the program starts, along with an option to exit the program.
- The user must put in some amount of money before an item can be selected.
- Only one item can be vended at a time.
- If the user selects an item that costs more than the amount the user put into the vending machine, the program should display a message indicating insufficient funds and then redisplay the amount the user had put into the machine.
- If the user selects an item that costs equal to or less than the amount of money that the user put in the vending machine, the program should display the change returned to the user. Change must be

displayed as the number of quarters, dimes, nickels, and pennies returned to the user.
- Vending machine items must be stored in a file. Inventory for the vending machine must be read from this file when the program starts and must be written out to this file just before the program exits. The program must track the following properties for each item:
  - Item name
  - Item cost
  - Number of items in inventory
- When an item is vended, the program must update the inventory level appropriately. If the machine runs out of an item, it should no longer be available as an option to the user. However, the items that have an inventory level of zero must still be read from and written to the inventory file and stored in memory.

Hint: to make change, create a Change class that takes the amount of change due to the user (in pennies) and then calculates the number of quarters, dimes, nickels, and pennies due back to the user. This class should have accessors for each of the coin types.

Guidelines

- You should take considerable time to design this application before you even think about writing code. Follow the Service Layer and DAO interface design approaches shown in the write-ups and videos.

Discuss your design with your mentor and get basic approval before you begin coding.

- This application must follow the MVC pattern used for all previous labs (App class, Controller, View, Service Layer, DAO) – this includes the use of constructor based dependency injection.
- You must have a full set of unit tests for your DAO and Service Layer components.
- You must use BigDecimal for all monetary calculations where applicable.
- You must use application specific exceptions and your application must fail gracefully under all conditions (i.e. no displaying a stack trace when an exception is thrown). At a minimum you should have the following application specific exceptions thrown by your Service Layer:
    - One that is thrown when the user tries to purchase an item but doesn't deposit enough money (i.e. `InsufficientFundsException`)
    - One that is thrown when the user tries to purchase an item that has zero inventory (i.e. `NoItemInventoryException`)
- Use enums to represent the values of different coins.
- Include an Audit DAO to log events and the time they occurred.

Submitting Your Assessment

Your instructor will provide instructions on how to submit your work to a GitHub classroom.

After submitting your work, schedule a time with a staff member to review your code. If you are attending the Guild in person, your code will be reviewed during the weekly code review.

Be prepared to answer questions about your code and thought processes when you complete the code review.

Vending Machine v2.4

Criteria	Ratings			Pts
Specifications Apprentice applies the specifications to the application.	**10.0 pts** **Meets Expectations**	**5.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	10.0 pts
Dependency Injection The application uses dependency injection.	**10.0 pts** **Meets Expectations**	**5.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	10.0 pts
MVC The application uses an MVC pattern.	**10.0 pts** **Meets Expectations**	**5.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	10.0 pts
I/O Operations The application can perform I/O operations to a file to store and retrieve data.	**10.0 pts** **Meets Expectations**	**5.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	10.0 pts
Use Service Layer The application has a service layer that contains business logic.	**10.0 pts** **Meets Expectations**	**5.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	10.0 pts

Criteria	Ratings			Pts
Unit Tests The application has a full set of unit tests for the DAO and service layer.	**10.0 pts** **Meets Expectations**	**5.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	10.0 pts
BigDecimal The application uses BigDecimal for all monetary calculations.	**5.0 pts** **Meets Expectations**	**3.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	5.0 pts
Lambdas and Streams The application uses lambdas and streams in at least part of the DAO.	**5.0 pts** **Meets Expectations**	**3.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	5.0 pts
Enums The application uses enums to represent coins and their values.	**5.0 pts** **Meets Expectations**	**3.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	5.0 pts
Audit Log The application includes an audit log using the Java DateTime API.	**5.0 pts** **Meets Expectations**	**3.0 pts** **Needs Improvement**	**0.0 pts** **No Submission**	5.0 pts

Criteria	Ratings			Pts
Explain Service Layer The apprentice can explain the role of the service layer.	**5.0 pts Meets Expectations**	**3.0 pts Needs Improvement**	**0.0 pts No Submission**	5.0 pts
Explain Unit Testing The apprentice can explain why we write unit tests.	**5.0 pts Meets Expectations**	**3.0 pts Needs Improvement**	**0.0 pts No Submission**	5.0 pts
Code Style Code is written with appropriate indents, naming conventions, and comments so that other developers can read the code easily.	**10.0 pts Meets Expectations**	**5.0 pts Needs Improvement**	**0.0 pts No Submission**	10.0 pts
				Total Points: 100.0