Project -1 Specification

Version : V1

Date: July 10, 2023, Sept/09/2023

Author: Naganand Kanagal

# Java Project

"We've got too much inventory in the back rooms and our processes are not where we want them to be and that's causing some undue shrinkage and some out-of-stocks," Foran said.

The company has already started to implement some changes to fix these issues, such as marking down foods that are nearing their expiration date to reduce the amount of food that goes to waste.

-Wal-Mart US CEO Greg Foran

Source: "https://www.businessinsider.com/wal-marts-ceo-reveals-8-main-problems-2015-4

## Business case:

This project theme is based on the above information that appeared on Business Insider a few years ago.

Design an application with the following features:

### Section A:

Store a minimum and a maximum number of items of the product that can be displayed on the shelf. As soon as the system notices that it has reached the threshold number/weight, it is going to send a notification to replenish the items and store this information in database with timestamp.

For example:

1. Number of fresh apples on the shelf needs to be between 10 lbs and 45 lbs.

2. The max number of lawn mowers of a given type that can be on display is 5 and the minimum is 2.

3. The maximum number of baby diapers pack of a type/brand that can be on display is 15, the minimum is 5.

### Section B:

Store the expiry date of each product. System determines if the product has reached expiry date by comparing it with the current date and sends a notification with the number of items and store the same information in database.

### Section C:

Store "Markdown Date", this is the duration of time from the expiry date that the product has to be marked down. System sends a notification stating the products that need to be marked down.
Store the same in database.

For this release, the application will run from the command line, though a future release will include a web front end (extra credit, not mandatory) and implementing with database (MySQl or MS Access)(extra credit, not mandatory).

This assignment will test your understanding of basic core Java concepts.

You will create a Java project with the following functions. The functions will be called via a text-based Command Line Interface (CLI), hence your program should be able to accept input from a basic CLI and display information to the console.

## General requirements for functions:

All functions either returns a result or displays the results in a readable format.

All functions display meaningful error messages on user error so that the user can take corrective actions.

All functions displays meaningful messages to the user if there is no output.

All functions should throw a relevant exception when it comes across an error or a validation fails.

The application will have the following methods:

**createProduct:** ProductID, ProductName, ExpiryDate, TimeDurationForMarkDown.

All of the arguments are mandatory.

1. Creates a product and stores in database.
2. The ProductID must be unique. If we try to create with a non-unique ProductID, display a message “ProductName should have a uniqueID, the ProductName already exists with the same uniqueID”.
3. If the system creates the product successfully, display, “ProductName with the ProductID created successfully".
4. If the createProduct command is called without parameters, display a helpful message that productID and ProductName are required. And that other arguments take default values.
5. Default values,
    a. ExpiryDate: 3 months from the day of entry.
    b. TimeDurationForMarkDown: 6 days before the ExpiryDate.
6. Log any other exceptions and display a helpful error message.

displayProduct: ProductName(optional), ProductID(optional).

1. The console/GUI should display the productName and all the other attributes.
2. If the product is not found, display, “Productname/ProductID notfound”
3. If the display_product command is called with no parameters, all products with product name and ProductID will be displayed.
4. Log any other error/exception with a user friendly (helpful) message.

## Functions specific to Section A of the requirements:

displayProductToRefill: none

1. Display all the products in the store that need to be replenished at that time and the product and the quantity that needs to be replenished.

2. If no products need to be replenished, log/display a relevant message.

displayProductToRefill: ProductID

1. If the product needs to be replenished, display the number of items/weight that needs to be replenished. Otherwise display a relevant message.

2. Validate the ProductID and display a relevant message if the validation fails.

displayProductCount:none

1. Logs/displays number/weight of all the products on the shelf.

2. If no products on shelf display/log a relevant message.

displayProductCount: ProductID

1. Logs/displays number/weight of that product on the shelf.

2. Validate the ProductID and display a relevant message if the validation fails.

## Functions specific to Section B of the requirements:

### displayProductsExpiryDate: productID

1. Displays/logs the expiry date of the product. ProductID is mandatory.
2. Validate the ProductID and display a relevant message if the validation fails.

### displayProductsExpiryDate: none

1. Displays all products and their expiry date.
2. If no products on shelf display/log a relevant message.

### displayExpiredProducts

1. Displays all expired products on the shelf with ProductName, ProductID and ExpiredDate. Make this message very noticeable.
2. If no products on shelf display/log a relevant message.

## Functions specific to Section C of the requirements:

### displayProductsInMarkDown:none

1. Display all products that are past the MarkdownDate.
2. If none display a relevant message.

### displayProductsForMarkDown:none

1. Display all products that need to be marked down a week from now.
2. If none display a relevant message.

EXTRA CREDIT – 20 points

Create a GUI front end (web or client). Instead of interfacing with your application via the command line, you will instead create a front end to access your application. If you do decide to go this route, we will use the GUI to test your functions.

## Submission Details

Create a fat jar and ZIP all of your source code and fat jar into a file for submission.

We should be able to compile your project from your source code , no errors will be tolerated.

## Submission will be graded as follows:

20 points for successful formatting and compilation.

40 points for successful operation against the test scenarios.

40 points for successful execution via fat jar.

Code will be executed from the fat jar. No errors/exceptions will be tolerated.

## Test Cases:

### createProduct: ProductID, ProductName, ExpiryDate, TimeDurationForMarkDown.

First two arguments are mandatory.

I. Run the above function with all the right arguments. It should display a message “ProductName with the ProductID created successfully".
II. Try to create another product with a duplicate ProductID. It should fail to create a product. Display a message “ProductName should have a uniqueID, the ProductName already exists with the same uniqueID”.
III. Run the createProduct command without parameters, display a helpful message that productID and ProductName are required. And that other arguments take default values.
IV. Make sure the default values are stored,
    a. ExpiryDate: 3 months from the day of entry.
    b. TimeDurationForMarkDown: 6 days before the ExpiryDate.

### displayProduct: ProductName(optional), ProductID(optional).

I. Run the function displayProduct(), the console/GUI should display the productName and all the other attributes.
II. If the product is not found, display, “Productname/ProductID notfound”
III. If the displayProduct command is called with no parameters, all products with product name and ProductID will be displayed.
IV. Log any other error/exception with a user friendly (helpful) message.

## Functions specific to Section A of the requirements:

### displayProductToRefill: none

1. Display all the products in the store that need to be replenished at that time and the product and the quantity that needs to be replenished.

2. If no products need to be replenished, log/display a relevant message.

### displayProductToRefill: ProductID

1. If the product needs to be replenished, display the number of items/weight that needs to be replenished.

2. If none display a relevant message.

3. Validate the ProductID and display a relevant message if the validation fails.

### displayProductCount:none

1. Logs/displays number/weight of all the products on the shelf.

2. If no products on shelf display/log a relevant message.

### displayProductCount: ProductID

1. Logs/displays number/weight of that prodcut on the shelf.
2. Validate the ProductID and display a relevant message if the validation fails.

## Functions specific to Section B of the requirements:

### displayProductsExpiryDate: productID

1. Displays/logs the expiry date of the product.
2. Validate the ProductID and display a relevant message if the validation fails.

### displayProductsExpiryDate: none

1. Displays all products and their expiry date.
2. If no products on shelf display/log a relevant message.

### displayExpiredProducts

1. Displays all expired products on the shelf with ProductName, ProductID and ExpiredDate. Make this message very noticeable.
2. If no products on shelf display/log a relevant message.

## Functions specific to Section C of the requirements:

### displayProductsInMarkDown:none

1. Display all products that are past the MarkdownDate.
2. If none display a relevant message.

### displayProductsForMarkDown:none

1. Display all products that need to be marked down a week from now.
2. If none display a relevant message.