Digikala Project Document

Advanced Programming - Spring 2023

Introduction

In this project, you will build an online shop simulator App providing e-commerce services, similar to Digikala or Amazon. The application should allow users to simulate a simplified version of the online shopping experience through the command line interface.

* Note that this is a solo project meant to be undertaken by each student individually.

Objective

A Major goal of this project for you, is to exercise your creativity and implement your own unique approach to building the application. No template code will be provided for this project as you are expected to build it from start to finish.

You are allowed to use any Java library to accomplish the project goals, but you are expected to demonstrate a strong understanding of core Java concepts such as inheritance, polymorphism, abstraction, and encapsulation.

Main Tasks

Your application should have a menu accessible in the command line interface. All interactions between the user and the app must be done through this interface, so be sure to provide a list of available commands in the terminal. The following functionalities MUST be implemented:

- **Shop**: Your shop must have a name, a web address, and a support phone number. You should also save a list of accounts, a list of products, a list of all orders, and the total profit gained by the shop.

- **Account Types:** You must implement a minimum of three different Account types with varying access levels. An Admin, a User, and a Seller Account type must be available, but you can add more. Remember to use OOP principles while implementing the Account system.

- **User Attributes:** A user must provide a username, a password, an email address, a phone number, and an address as attributes. Your application should also save a shopping cart, a list of orders, a list of purchased products, and a wallet for each user as well. Additionally, each user must have a profile screen that can be viewed by the admins.

- **Registration:** Users must be able to register an account and log in to access the shopping functionalities. Allow users to edit their personal information.

- **Product Catalog:** Users should be able to search for products, view product details, and add products to their cart.

- **Admin Attributes:** An admin must have a username, a password, and an email address. New admins can only be added by other admins. Your program must have at least one predefined admin.

- **Seller Attributes:** A seller must provide a company name, a password, a list of available products, and a wallet. Sellers must first get authorization from an admin before being allowed to sell products. Sellers should be able to add new products to their list by providing the necessary info.

- **Wallet:** Only admins should be able to add funds to a user's wallet. Users can send a request to the admins for adding funds. Funds are only added to a seller's wallet after order confirmation.

- **Product:** Each product must have a name, a price, quantity, a list of comments, and additional data based on product type and category. Users can add a number of each product to their cart, but this number cannot be greater than the current quantity of it.

- **Category:** You must define multiple categories for the products available (minimum of 10) such as electronics, clothes, books, etc. Be sure to use inheritance to create a proper structure for categories and subcategories.

- **Cart Management:** Users should be able to view their cart, remove items from their cart, and update the number of items in their cart.
- **Order:** Users should be able to checkout their cart and receive a confirmation of their order. Each order must have a date, a total price, and the details of the buyer and seller entities. Your application must save a list of all orders. Once an order is submitted, the remaining stock of all purchased products must be updated.
- **Checkout Process:** Each order must first be confirmed by an admin, before a transaction occurs. If the user doesn't have enough funds in their wallet, the order will be canceled. Whenever an order is confirmed, the total cost will be removed from the user's wallet and a portion of it added to the seller's wallet. The shop's cut is 10% of each product's price.
- **Assign ID:** Some of the entities you are using must be uniquely identifiable from each other. **It is up to you to determine which classes need an ID attribute and to implement a proper ID for each instance of them**.
- **Data Management:** You will be working with a variety of data in your application, it's up to you to find an efficient way to organize the data throughout your project. Use appropriate data types and data structures. Try to avoid data redundancy as much as possible.

Notes

- Make sure to use a Package Manager for your project (either Maven or Gradle)
- Create a new Git branch "Develop". Any additional branches you create should be named appropriately and merged into Develop. Do not delete old branches after merging them.
- You must override certain methods such as toString() and equals() in some of the classes you implement.
- Use Design Patterns - You should use appropriate design patterns to ensure your code is modular and maintainable.
- You must design unit tests for different stages of your project.

Report

You are expected to write a detailed report on the process you went through while working on your project. Include the following details in your report:

- A UML diagram for each class
- How you determined which entities need to be uniquely identifiable
- Challenges and bugs that you faced and how you managed to solve them
- Explanation regarding any additional feature you implemented

Bonus Tasks

Below you can find a collection of Additional features you can implement in your project. While this is our recommended list, you are free to add more optional and challenging features at your discretion. In other words, **your creative effort won't go unnoticed.**

* Remember to mention any bonus task you finish in your report, and provide an adequate explanation with at least one test for each one.

1. Ensure the data used by your application is persistent. You can use simple file systems to achieve this or utilize a database system such as SQL or MongoDB.
2. Design a GUI (Graphical User Interface) for your application using **JavaFX**. The GUI must offer as many options as the CLI version of the project. Using images and icons or adding animations is optional.
3. Implement a more complex search system:
    - Display product names that only contain a section of the searched word
    - Display product names that contain more than just the searched word
4. Implement a product rating system and display the rating. Users should only be allowed to rate a product once, after which they can edit the rating.
5. Implement a product recommendation system based on the user's search history and the overall product rating (if you've implemented it).
6. Display two recommended products while viewing a product's page. One of the recommended products should be more expensive than the main product and the other should be less expensive (unless such products don't exist).

7. Allow more than one seller to have a product available for sale (different prices). On the product page, display the different seller options for each product, with their respective price.
8. Calculate a shipping fee for each order based on the user's location.
9. Create a Notification list for users and sellers to keep track of their transactions.
10. Implement a subscription service that offers free shipping and special discounts for a monthly fee.
11. Implement a refund system. Users should be able to request a refund for a product they ordered, and every refund request must be confirmed by an admin.
12. Create a ledger system for sellers that can keep track of their sales records.

Tools & Packages

You can find a selection of helpful libraries, apps, and tools here. You are not required to use any of them, but they may prove useful in your project.

- UUID: The Universally Unique Identifier class can be used to generate IDs for objects that need to be uniquely identified. Note that using a UUID for every class might not always be the best approach.
- Scene Builder: A visual layout tool that allows developers to design a JavaFX UI by dragging and dropping components onto a scene, and generating FXML markup code that can be used in Java applications.
- PostgreSQL: A powerful open-source object-relational database management system that provides robust data integrity and advanced query capabilities.

Evaluation

- Your code should compile and run without any errors.
- Your code should be well-organized, readable, properly commented, and should follow clean code principles. Use appropriate Java naming conventions.
- Your code should follow OOP principles. Concepts such as polymorphism and inheritance should be efficiently used in your code.
- You should use Git for version control and include meaningful commit messages. Utilize as many branches as necessary.
- You will be required to present your code to a team of judges at a later date for the final evaluation.

Submission

- Push your code to your fork on GitHub
- Upload your report to your fork
- Merge your “Develop” branch with the main branch of your fork

The deadline for submitting your project is **Wednesday, April 5 (16th of Farvardin)**. Any commit made after this date will not affect your score.

Good luck and happy coding!