**Date: 11/07/2024**

**Report: e-Pharmacy Management System By: Afridi Akbar Ifty**

**<Contact : https://iamafrididev.netlify.app>**

**1. Introduction**

The **e-Pharmacy Management System** is designed to automate the management of pharmaceutical operations, including stock management, medication search, customer purchases, and order tracking. The system allows users (such as customers and pharmacists) to search for medications, order them, and manage inventory. The system is implemented using Object-Oriented Programming (OOP) principles, including inheritance, abstraction, and interfaces.

**2. System Overview**

The system has the following components:

1. **Medication Class**: Represents a medication with details such as name, manufacturer, batch number, stock quantity, expiry date, and price.
2. **PharmacyUser Abstract Class**: Serves as a base class for all types of users in the system (customers and pharmacists).
3. **Customer and Pharmacist Classes**: Implement the PharmacyUser class. The Customer class allows users to purchase medications, while the Pharmacist class manages the stock.
4. **Searchable Interface**: Provides a standard way for searching medications in the pharmacy.
5. **Pharmacy Class**: Manages the collection of medications and manages operations like adding, searching, and displaying medications.
6. **Main Class**: Acts as the entry point of the program and contains the interactive user interface for managing the pharmacy system.

**3. Class Design**

1. **Medication Class**:
   * This class contains attributes like name, manufacturer, expiryDate, batchNumber, stockQuantity, and price. Methods include getters and setters for these attributes, along with a constructor to initialize them.
   * The class allows tracking of stock levels and medication prices.

‘

Example:

public class Medication {

private String name;

private String manufacturer;

private String expiryDate;

private String batchNumber;

private int stockQuantity;

private double price;

// Constructor and getters/setters

}

1. **PharmacyUser Abstract Class**:
   * This class serves as a base class for both Customer and Pharmacist. It contains common attributes such as name, userId, and userType (e.g., customer or pharmacist), and methods for retrieving these attributes.
2. **Customer Class**:
   * Inherits from PharmacyUser. The Customer class allows customers to purchase medications. The purchaseMedication method checks if a customer can afford the medication and updates the stock accordingly.
3. **Pharmacist Class**:
   * Inherits from PharmacyUser. The Pharmacist class allows the pharmacist to restock medications in the pharmacy by updating stock quantities.
4. **Searchable Interface**:
   * The Searchable interface defines a search method, which is implemented by the Pharmacy class. It allows users to search for medications based on keywords (e.g., name, manufacturer).
5. **Pharmacy Class**:
   * This class manages the pharmacy’s medications in a List<Medication>. It provides methods to add, search, and display medications, as well as to check if a medication exists and has sufficient stock.
6. **Main Class**:
   * This is the entry point for the program. It displays a menu that allows users to interact with the system. The menu includes options for searching medications, viewing pharmacy stock, ordering medications, and exiting the system.

**4. Workflow and Features**

1. **Menu System**: The system starts with a menu that offers the following options:
   * **Search for Medication**: Allows users to search medications by name or manufacturer.
   * **View Pharmacy Stock**: Displays all available medications in the pharmacy in a table format.
   * **Order Medication**: Enables customers to purchase medications, specifying the quantity and processing the payment.
   * **Exit**: Exits the system.
2. **Medication Search**: When searching for medications, the system allows users to enter a keyword (either the name or manufacturer). The system then displays matching medications, and users are prompted to purchase the medication.
3. **Stock Management**: Pharmacists can manage stock by restocking medications. The system allows updating the stock quantities of medications.
4. **Purchase Flow**: Customers can order medications. The system checks if the user has enough balance to complete the purchase. Once the purchase is completed, the total bill is displayed, and the user is prompted to order more or exit.
5. **Total Bill Calculation**: After ordering medications, the system calculates the total cost based on the price and quantity of the medication. This total is displayed to the user.

**5. Exception Handling**

* **MedicationNotFoundException**: This exception is thrown when a medication is not found in the pharmacy during a search or order attempt.
* **InvalidMedicationException**: This exception is thrown when trying to add a medication with invalid details (e.g., empty name or manufacturer).
* **InsufficientStockException**: This exception is thrown when attempting to purchase more stock than is available in the pharmacy.

**6. UML Diagram Explanation**

The **UML diagram** visually represents the structure of the e-Pharmacy Management System:

* **Medication Class**: Contains attributes related to medications and methods for managing them.
* **PharmacyUser Class**: The base class for all users (Customer, Pharmacist), holding common attributes.
* **Customer & Pharmacist Classes**: Inherit from PharmacyUser, with Customer having methods for purchasing and Pharmacist for restocking.
* **Searchable Interface**: Provides the contract for searching medications in the pharmacy.
* **Pharmacy Class**: Manages the medications and supports search and stock management.
* **Main Class**: The entry point of the program where users interact with the system.

**7. Conclusion**

The e-Pharmacy Management System effectively uses OOP principles such as inheritance, encapsulation, and abstraction to model the behavior of a pharmacy, its users, and medications. The system provides essential functionalities such as searching for medications, managing stock, and ordering medications, while also handling exceptions gracefully to ensure smooth user experience. The addition of a user-friendly menu and detailed transaction management ensures that customers and pharmacists can interact with the system efficiently.

This system can be extended to include features such as customer login, pharmacy reports, and integration with external databases or payment systems.