

# Inventory Management System

## Project Overview:

You are tasked with developing a comprehensive Inventory Management System for a small store. This system should manage various types of items, handle payments, and process orders. Implement interfaces, abstract classes, and multiple concrete classes.

## Requirements:

### Interfaces and Abstract Classes

#### 1. Item Interface (15 points):

- Create an Item interface to represent items in the inventory.
- Define methods for getting item details, calculating value, and displaying the item's description.

#### 2. Categorizable Interface (10 points):

- Create a Categorizable interface that represents items that can be categorized.
- Include methods for setting and getting the item category.

#### 3. Breakable Interface (10 points):

- Create a Breakable interface to indicate items that can break.
- Include methods for checking if an item is breakable and for handling item breakage.

#### 4. Perishable Interface (10 points):

- Create a Perishable interface to represent items that can perish.
- Include methods for checking if an item is perishable and for handling item expiration.

#### 5. Sellable Interface (10 points):

- Create a Sellable interface to represent items that can be sold.
- Include methods for setting and getting item prices.

#### 6. \*Abstract Item Class (15 points):

- Create an abstract class AbstractItem that implements the Item, Categorizable, Breakable, Perishable, and Sellable interfaces.
- Implement common functionality such as getting item details.



- Provide default implementations for category, breakable, perishable, and sellable attributes.

## Superclasses and Inheritance

### 7. Inventory Superclass (20 points):

- Create an InventoryItem superclass \*(that extends AbstractItem).
- Add instance variables for item ID and quantity.
- Implement getters and setters for ID and quantity.

### 8. Item Types (30 points):

- Create subclasses for specific item types like ElectronicsItem, GroceryItem, and FragileItem that inherit from InventoryItem.
- Implement constructors for these subclasses to set specific attributes like weight for fragile items.
- Override relevant methods to calculate item values differently for each type.

## File I/O, User Interface, Payments, and Orders

### 9. File I/O (15 points):

- Implement methods to save and load inventory data to/from text files.
- Use a well-defined file format for data storage.

### 10. User Interface (15 points):

- Create a command-line interface (CLI) to interact with the inventory system.
- Allow users to add items, remove items by ID, display a list of items, categorize items, and place orders.
- Display a menu for user choices and handle user input gracefully.

### 11. Payments and Orders (20 points):

- Implement classes for Payment and Order.
- Allow users to create orders, calculate order totals, and process payments.
- Update inventory quantities after orders are placed.

## Error Handling and Documentation

### 12. Error Handling (10 points):

- Implement robust error handling to address potential issues, such as invalid user input, file I/O errors, and handling exceptions properly.

### 13. Documentation (10 points):



- Document your code with meaningful comments and explanations.
- Include comments explaining the purpose and usage of each class, interface, and method.

**Extra Credit (10 points):**

- Implement additional features, such as searching for items by name or category, updating item quantities, sorting items by name, category, or price, or implementing discount codes for orders.

**Example Usage of the Application:**

1. Add Items to Inventory:
  - Add to add a new item.
  - Specify item details, including name, category, price, quantity, and item type.
2. List Inventory Items:
  - List to display a list of all inventory items.
3. Categorize Items:
  - Categorize to categorize items based on their type or category.
4. Place an Order:
  - Order to start the order process.
  - Add items to the order and specify quantities.
  - Calculate the total cost and apply payments.
5. Error Handling:
  - Ensure that the application handles invalid input, out-of-stock items, and file I/O errors gracefully.

**E-commerce Console Application**

## Project Overview:

You are tasked with developing an E-commerce Console Application that includes inventory management, order processing, and payment handling. This project builds upon the existing inventory management system and introduces payment processing capabilities.

## Requirements:

### Part 1: Inventory Management (Same as Previous Assignment)

### Part 2: Payment Processing

#### 1. Payment Processor (20 points):

- Create a **PaymentProcessor** class to handle payments.
- Implement methods for processing payments using various payment methods (e.g., credit card, PayPal).
- Include validation for payment methods and simulate payment authorization.

#### 2. Payment Methods (25 points):

- Create interfaces or abstract classes for different payment methods such as **CreditCardPayment**, **PayPalPayment**, etc.
- Implement payment methods using appropriate attributes (e.g., card number, PayPal account) and validation.
- Use interfaces or abstract classes to ensure consistent payment processing.

### Part 3: User Interface Enhancement

#### 3. E-commerce Console Interface (20 points):

- Enhance the console interface to allow users to select and purchase items.
- Implement the shopping cart functionality to add items to the cart, view the cart, and place orders.
- Integrate payment processing into the ordering process.

### Part 4: Order Processing

#### 4. Order Class (15 points):

- Create an **Order** class to represent orders.
- Include details such as order ID, items, quantities, total cost, and payment method.



- Implement methods for calculating the order total, processing payments, and updating inventory quantities.

## Part 5: Error Handling and Documentation

### 5. Error Handling (10 points):

- Implement robust error handling to address potential issues, such as invalid user input, out-of-stock items, payment authorization errors, and file I/O errors.

### 6. Documentation (10 points):

- Document your code with meaningful comments and explanations.
- Include comments explaining the purpose and usage of each class, interface, and method.

### Example Usage of the Application:

```
import java.util.Scanner;

public class ECommerceApp {

    public static void main(String[] args) {

        InventoryManager inventoryManager = new InventoryManager();
        PaymentProcessor paymentProcessor = new PaymentProcessor();
        Scanner scanner = new Scanner(System.in);
        System.out.println("Welcome to the E-commerce Console Application!");
        displayMenu();
        boolean isRunning = true;
        while (isRunning) {
            System.out.print("Enter command (1-4): ");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    inventoryManager.listItems();
                    break;
                case 2:
                    // ...
                default:
                    System.out.println("Invalid command. Please try again.");
            }
        }

        scanner.close();
    }
}
```



```
}  
  
private static void displayMenu() {  
    System.out.println("Menu:");  
    System.out.println("1. List Items");  
    // .....  
}  
}
```

Good luck and have fun!