Online Shopping Cart System

Objective:
Design and implement an Online Shopping Cart System that allows customers to browse products, add items to their cart, apply discounts, and checkout, using object-oriented principles, with a focus on classes and inheritance.

Problem Description:

The Online Shopping Cart System will manage products, handle customer interactions with the cart, and process orders. The system will be designed using object-oriented principles, with a focus on using classes and inheritance to represent different entities in the shopping process.

Key Components:

1. Base Class: `Product`
   - Attributes:
     - `productID`: A unique identifier for the product.
     - `name`: The name of the product.
     - `price`: The price of the product.
     - `quantity`: The number of units available in stock.
   - Methods:
     - `getPrice()`: Returns the price of the product.
     - `getProductInfo()`: Displays the product's details.
     - `reduceStock(int quantity)`: Reduces the stock of the product by the specified quantity.
   - Description: The `Product` class serves as a base class representing any product in the store. It provides common attributes and methods that can be inherited by more specific product classes.

2. Derived Classes:
   - `Electronics` Class (inherits from `Product`):
     - Additional Attributes:
       - `brand`: The brand of the electronic product.
       - `warrantyPeriod`: The warranty period in years.
     - Additional Methods:
       - `displayElectronicsInfo()`: Displays detailed information about the electronic product, including brand and warranty period.
     - Description: The `Electronics` class extends `Product` and includes attributes and methods specific to electronic products.

   - `Clothing` Class (inherits from `Product`):
     - Additional Attributes:
       - `size`: The size of the clothing item (e.g., S, M, L).
       - `color`: The color of the clothing item.
     - Additional Methods:

- `displayClothingInfo()`: Displays detailed information about the clothing item, including size and color.
- Description: The `Clothing` class extends `Product` and includes attributes and methods specific to clothing items.

- `Book` Class (inherits from `Product`):
  - Additional Attributes:
    - `author`: The author of the book.
    - `publisher`: The publisher of the book.
  - Additional Methods:
    - `displayBookInfo()`: Displays detailed information about the book, including author and publisher.
  - Description: The `Book` class extends `Product` and includes attributes and methods specific to books.

3. Shopping Cart Class:
- Attributes:
  - `cartItems`: A collection of `CartItem` objects, where each `CartItem` represents a product added to the cart and its quantity.
- Methods:
  - `addItem(Product product, int quantity)`: Adds a specified quantity of a product to the cart.
  - `removeItem(Product product)`: Removes a product from the cart.
  - `calculateTotal()`: Calculates the total price of all items in the cart.
  - `applyDiscount(Discount discount)`: Applies a discount to the total price.
  - `checkout()`: Processes the payment and finalizes the purchase.
- Description: The `ShoppingCart` class manages the products added to the cart, calculating the total cost, applying discounts, and facilitating the checkout process.

4. Discount Class:
- Attributes:
  - `discountCode`: A unique code representing the discount.
  - `discountPercentage`: The percentage of the discount to be applied.
- Methods:
  - `applyDiscount(double totalAmount)`: Applies the discount to the total amount and returns the discounted total.
- Description: The `Discount` class represents a discount that can be applied to the total price of the items in the shopping cart. It interacts with the `ShoppingCart` class to provide price reductions.

5. Customer Class:
- Attributes:
  - `customerID`: A unique identifier for the customer.
  - `name`: The name of the customer.
  - `email`: The email address of the customer.
  - `shoppingCart`: An instance of `ShoppingCart` associated with the customer.
- Methods:

- `addItemToCart(Product product, int quantity)`: Adds an item to the customer’s shopping cart.
- `removeItemFromCart(Product product)`: Removes an item from the customer’s shopping cart.
- `checkoutCart()`: Finalizes the purchase by checking out the shopping cart.
- Description: The `Customer` class represents a customer who interacts with the shopping cart, adding and removing items and finalizing purchases.

6. Online Store Class:
- Attributes:
  - `products`: A collection of all products available in the store.
  - `customers`: A collection of all registered customers.
- Methods:
  - `addProduct(Product product)`: Adds a new product to the store's inventory.
  - `registerCustomer(Customer customer)`: Registers a new customer.
  - `findProductById(String productId)`: Searches for a product by its ID and returns it if found.
  - `listAvailableProducts()`: Lists all products currently available in the store.
- Description: The `OnlineStore` class manages the inventory of products and the customers registered with the store. It facilitates product browsing, customer registration, and interaction with the shopping cart.

7. Online Shopping System Launcher:
- Purpose: Implement an `OnlineShoppingLauncher` class with a `main()` method to simulate the shopping process. This class will:
  - Create instances of `Product` subclasses (like `Electronics`, `Clothing`, `Book`).
  - Add products to the store.
  - Register customers.
  - Simulate customers adding products to their carts, applying discounts, and checking out.
  - Display the details of transactions, including product information and final prices.

Example Usage:

- Create a `Product` base class with common attributes and methods, and extend it with specific product types like `Electronics`, `Clothing`, and `Book`.
- Develop a `ShoppingCart` class to manage the process of adding, removing, and purchasing products, and applying discounts.
- Implement a `Customer` class that interacts with the shopping cart, facilitating user actions like adding items and checking out.
- Simulate the online shopping experience using a launcher class to integrate all components and demonstrate a complete transaction from product selection to checkout.

This problem statement emphasizes the use of inheritance to manage different types of products while promoting code reuse and scalability. It encourages the application of object-oriented principles to build a comprehensive online shopping cart system in Java.