Project Report - Shopping Cart System

Project Description:

The Shopping Cart System is a Python application developed to manage a shopping cart for customers, similar to those found in online stores. The system comprises three main classes - Product, ShoppingCart, and two subclasses of Customer, namely LoyalCustomer and BargainHunter. The Product class represents individual items available in the store, with attributes such as name and price. The ShoppingCart class allows customers to add or remove products and keeps track of the items in their carts. The LoyalCustomer class is for customers who can buy exclusive products, and the BargainHunter class is for customers who can buy any other products listed from low to high price.

Class Descriptions:

1. Product:
- __init__(self, name, price): Initializes a new Product object with the given name and price.
- __str__(self): Returns a string representation of the product, including its name and price.

2. ShoppingCart:
- __init__(self): Initializes an empty shopping cart.
- add_product(self, product, quantity=1): Adds a product to the shopping cart or increases its quantity if already present.
- remove_product(self, product, quantity=1): Removes a product from the shopping cart or decreases its quantity if present.
- __str__(self): Returns a string representation of the shopping cart, displaying the items and their quantities.

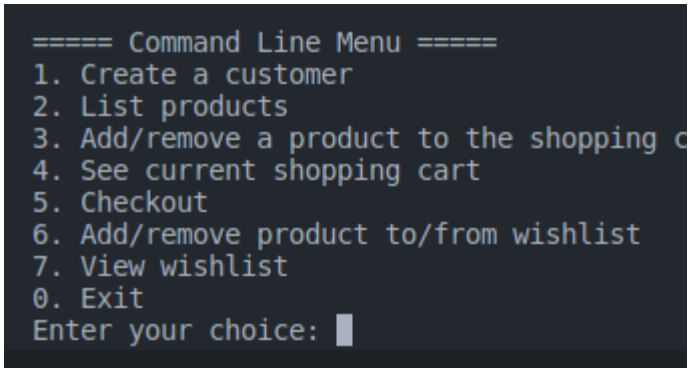
3. Customer (base class):
- __init__(self, name): Initializes a new Customer object with the given name.

4. LoyalCustomer (subclass of Customer):
- __init__(self, name): Initializes a new LoyalCustomer object with the given name.
- add_exclusive_product(self, product, quantity=1): Adds an exclusive product to the customer's list or increases its quantity.
- remove_exclusive_product(self, product, quantity=1): Removes an exclusive product from the customer's list or decreases its quantity.
- __str__(self): Returns a string representation of the loyal customer, displaying their name and exclusive products.

5. BargainHunter (subclass of Customer):
- __init__(self, name): Initializes a new BargainHunter object with the given name.
- add_to_cart(self, product, quantity=1): Adds a product to the shopping cart.
- remove_from_cart(self, product, quantity=1): Removes a product from the shopping cart.
- __str__(self): Returns a string representation of the bargain hunter, displaying their name and the shopping cart.

User Manual:

To use the Shopping Cart System, follow these steps:



1. Create a new customer by selecting option 1 from the command-line menu.
2. Once the customer is created, they can view the available products (option 2) and add/remove products to/from their shopping cart (option 3).
3. BargainHunter customers can view their shopping cart by selecting option 4. For LoyalCustomers, they can view their exclusive products and quantities.
4. Once the customer adds at least one product to the shopping cart, they can proceed with checkout (option 5) after confirming their cart's content and total amount.
5. Confirm the checkout, and the system will display a thank-you message and exit.

Additional Features

6. The user can add products to a wish list.
7. User can view or remove items from their wish list once added.

Difficulties and Challenging Parts:

The most challenging part of the project was designing the class hierarchy and implementing the correct logic to handle exclusive products for LoyalCustomers and the shopping cart for BargainHunters. It involved managing the interactions between the classes and ensuring that the system accurately represents a real-world shopping cart.

Overall, the Shopping Cart System was an enriching experience, providing insights into object-oriented programming and class design.