Project Report

Daniel Kohlbek

Description

The purpose of this project was to create and simulate a shopping cart system for an online store. You'll find 3 different Python files included in this project, **classes.py, createProducts.py, and main.py.** In classes.py are all the class definitions that were used in this project. The main classes are **Product, Cart, User, and Discount.** The **Product** class and its child classes, **DigitalProduct** and **PhysicalProduct** were responsible for creating objects of different products available in the online store. The **Cart** class offered common functionality like adding and removing products that you would typically find in a cart of an online shop. The **User** class creates user objects with their own unique **Cart** objects to go with and abstracts much of the functionality associated with the **Cart** class to make for a better user experience.

In **createProducts.py**, I wrote code to read in information from .csv files to create objects of products to be available in the online store.

In **main.py,** all the testing for each function of the project takes place.

Structure

Product class creates product objects and acts as the base class for DigitalProduct and PhysicalProduct. Contains methods update_quantity() to update the quantity of an item and get_product_info() that displays all the attributes of the product in the terminal.

Product
+product_id: int +name: str +price: float +quantity: int
+update_quantity(new_quantity: int): void +get_product_info(): void

DigitalProduct is a child class of Product that creates digital product objects that you could find in an online store. Attributes specific to this class are file_size and download_link. The function get_product_info() is overridden to include all the attributes of this class.

DigitalProduct: Extends Product
+product_id: int +name: str +price: float +quantity: int +file_size: int +download_link: str
+get_product_info(): void

PhysicalProduct is a child class of Product that creates physical product objects that you could find in an online store that require shipping. Attributes specific to this class are weight, dimensions, and shipping_cost. The function get_product_info() is overridden to include all the attributes of this class.

PhysicalProduct: Extends Product
+product_id: int +name: str +price: float +quantity: int +weight: float +dimensions: str +shipping_cost: float
+get_product_info(): void

Cart is a class that creates cart objects with useful functionality. It has one attribute cart_items which is a list of Product objects selected by a user. The functions add_product(), remove_product(), view_cart(), and calculate_total() are common things one needs to do during online shopping. This class also includes its own version of discount functions to apply to products.

Cart
-cart_items: list
+add_product(product: Product): void +remove_product(product_id: int): void +view_cart(): void +calculate_total(): float +percent_discount(product_id: int, percentage: float): void +fixed_discount(product_id: int, amount: float): void

User is a class that creates objects of a specified user. Its attributes are user_id, name (name of the user), and cart—a Cart object. This class provides abstraction of Cart functionality through the use of add_to_cart(), remove_from_cart(), and checkout() allowing for an easier user experience.

User
+user_id: int +name: str +cart: Cart
+add_to_cart(product: Product): void +remove_from_cart(product_id: int): void +checkout(): float

Discount is an abstract class that acts as a blueprint for other classes described below

Discount: Abstract Class
none
+apply_discount(total_amount: float): void

PercentageDiscount is a child class of Discount and can be used to calculate percent discount on select items.

PercentageDiscount: Extends Discount
+percentage: float
+apply_discount(total_amount: float): float

FixedAmount discount is a child class of Discount and can be used to calculate fixed discounts on select items.

FixedAmountDiscount: Extends Discount
+amount: float
+apply_discount(total_amount: float): float

Instructions

The 2 main classes you should be using as a user of this project are the **User** and **Product** classes along with child classes of **Product**. To create products, you should be using one of the **Product** classes outlined in **classes.py**. The reason why you are only using the **User** class and not **Cart** is because **User** objects are instantiated with a **Cart** class as an attribute and offers abstraction from **Cart** functionality. Instantiate your user with a user_id and your name, and a **Cart** object if you wish. From here you can use **add_to_cart()** and **remove_from_cart()** to add or remove items from your cart. Finally, you can use **checkout()** to calculate the total of your products, apply discounts, and clear your cart.

Verification of Sanity of Code

Screenshot of the creation of products, adding the products to user1 and user2 carts, and running view_cart():

```
# create instances of DigitalProduct
music_video = DigitalProduct(
    20, "Music Video", 14.99, 100, 2048, "mymusiclink.com")
album = DigitalProduct(21, "Music Album", 30.99, 150, 1024, "myalbum.com")

# create instances of PhysicalProduct
desk = PhysicalProduct(22, "Computer Desk", 75.99, 50, 40, "6 x 6", 15.00)
computer = PhysicalProduct(
    23, "Apple Computer", 899.99, 200, 10, "1 x 1", 10.00)
folder = PhysicalProduct(24, "School Folder", 5.99, 200, 1, "1 x .8", 2.00)

# create user1 and add digital products
user1 = User(10, "Daniel", cart=Cart(cart_items=[]))
user1.add_to_cart(music_video)
user1.add_to_cart(album)

# create user2 and add physical products
user2 = User(11, "Tim", cart=Cart(cart_items=[]))
user2.add_to_cart(desk)
user2.add_to_cart(computer)
user2.add_to_cart(folder)

# verify each users' cart
print("User1 cart")
user1.cart.view_cart()

print("\nUser2 cart")
user2.cart.view_cart()
```

Terminal Output:

```
User1 cart
***ITEMS IN CART***
Music Video
Music Album

User2 cart
***ITEMS IN CART***
Computer Desk
Apple Computer
School Folder
```

Screenshot of the creation of discounts, applying discounts to users' carts, and running checkout() on each cart then displaying the empty cart in the terminal:

```
# Create PercentageDiscount instance and apply to user1 cart
album_discount = PercentageDiscount(50)
user1.cart.percent_discount(21, album_discount)

# Create FixedAountDiscount and apply to user2 cart
computer_discount = FixedAmountDiscount(100)
user2.cart.fixed_discount(23, computer_discount)

# Run checkout() on user 1
print()
print(user1.checkout())
# Ensure cart is cleared
print(user1.cart.view_cart())

# Run checkout() on user 2
print()
print(user2.checkout())
# Ensure cart is cleared
print(user2.cart.view_cart())
```

Terminal output:

```
30.48
***ITEMS IN CART***
Your cart is empty.
None

881.97
***ITEMS IN CART***
Your cart is empty.
None
```

We can verify the discounts worked by looking at the totals. In user1's cart, the total would have been $45.98 without the discount. In user2's cart, the total would have been $981.97 without the discount.

Conclusion

The main challenge I faced while working on this project was figuring out the best way to implement discounts. Originally, I wanted to create a global dictionary with available discounts on specific items. However, after trying to implement this, I realized this method didn't fit within the project requirements, so I opted to manually create and apply discounts in my program instead. Another challenge I tackled was figuring out how to implement new Products into the program. The project only required me to manually create products in **main.py**, however I wanted to see if there was a way to import product information from an outside csv file, which I implemented in **createProducts.py**. I thought this might be worth trying since in a real-world scenario, it may be more common that you will have a list of products that need to be added to an online store that would take too long to manually write code for each product class. However, I realize a more efficient implementation of this would be to some sort of SQL database, but that is outside the scope of this project. Even though I didn't end up using it in **main.py**, it was worth learning how to read information from .csv files and dynamically create class objects. Overall, this project was a good way for me to learn more about OOP and gave me an idea of some real-world problem solving that can help me in my career. For future work, I would like to find a better implementation for discounts in the program and perhaps attach some sort of database to it for scaling.