Camilo Zapata

Project Report

Description of the project

The main purpose of this project is to create a simple shopping cart system that allows users to add products or remove products in their carts, apply percentage or fixed amount discounts and being able to checkout. The system allows for two different options for products such as Digital products that requests additional info like the products file size and digital link. Physical products being the other product that requires addition information such as weight, dimensions and shipping cost. The overall main goal was to use object-oriented programming using different classes and objects, as well as using different principles such as abstraction, inheritance, polymorphism and encapsulation.

Structure

Class Product
Attributes: product_id, name, price, and quantity

Update_quantity method
1 argument: New_quantity equals to old quantity

Get_product_info method
No argument; prints out the product info

Class Digital(Product)
Attributes product_id, name, price, quantity, file_size and download_link

Update_quantity method:

1 argument; new_quantity equals to old quantity

Get_product_info method
No argument; prints out the product info

Class PhysicalProduct(Product)
Attributes product_id, name, price, quantity, weight, dimensions, and shipping_cost

Update_quantity method:
1 argument; new_quantity equals to old quantity

Get_product_info method
No argument; prints out the product info

Class Cart
No argument; empty cart list

Add_product method
1 argument; append product to list

remove_product method
1 argument; remove product to list

View_cart method
No arguments; prints out product info in cart

Calculate_cart method

No arguments; prints out total price of cart

Clear_cart method
No arguments; clears cart

Class User
3 attributes; user_id, name, cart

Add_product method
1 argument; append product to user’s list

remove_product method
1 argument; remove product from user’s list

Checkout method
1 argument: discount can be applied to cart

Class Discount
1 class attribute; Total

Class PercentageDiscount
1 attribute; percentage

Apply_discount method
1 argument; total - (total*(percentage /100)

Class FIxedAmountDiscount
1 attribute; amount

Apply_discount method
1 argument; total_amount

<u>Summary of classes</u>

-Product class with basic attributes product id, name, price, and quantity. Two methods including update_qunatity() and get_product_info.

-DigitalProduct class that inherits the atrributes from Product class and has two additional ones file_size and download_link. Includes another method that overides the get_product_info to inlude additional information of the other two atributes that were added.

-PhysicalProduct class that inherits the attributes from Product class. Includes addition attrbutes weight, dimensions and shipping_cost. Same as the DigitalPrdoduct class that uses inheritance to provide addtional info from the method get_product_info

-Cart Class that has different methods for addding products to cart, removing products, viewing what products you have in car and a method to calculate the total. User class that has atrributes user_id, name, cart and methods for adding, removing, viewing products and checkout by delegating the cart class to perform those methods.

-Discount class that is an abstract class and implements apply_discount method that is used in the subclass FixedAmountDiscount and PercentageDiscount.

-PercentageDiscount class has method apply_discount to apply percentage based discounts to the total price.

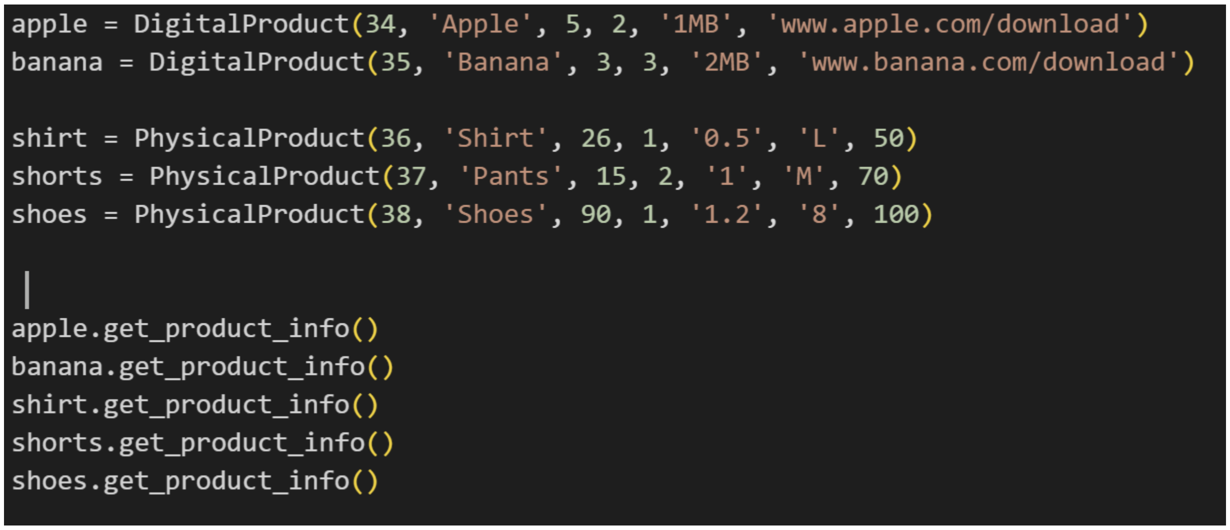
-FixedAmountDiscount class that has method apply_discount to apply fixed amount discount to the total price

3. Instructions Create objects from Digitalproduct or Physicalproduct including the correct attributes for each product like basic attributes like name, product_id, quantity, price and any additional attributes from the class of your choosing. Then create a user with its own user id, name and cart(). Then add products using

add_to_cart() with the name of the user then .add_to_cart(), with whatever product you want inside the parenthesis. Example Camilo.add_to_cart(apple). Then use view_cart() to display the list of products added in cart. If you want to apply discounts use either FixedAmountDiscount or PercentageDiscount with whatever value in the parenthesis. Then call checkout() to get the total price.

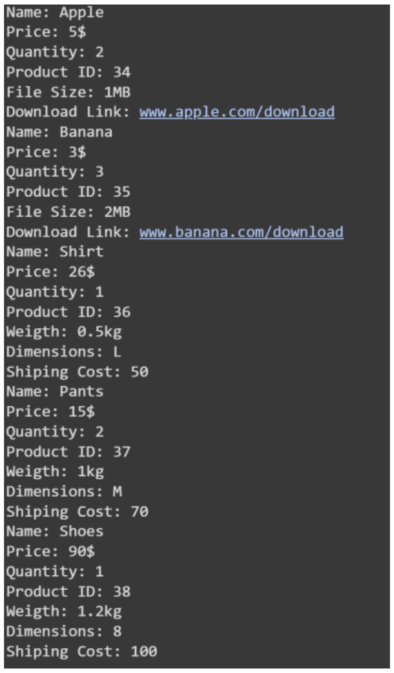
<u>Verification of sanity of code</u>

1. Create 2 instances of DigitalProduct and 3 instances of PhysicalProduct with appropriate attributes.



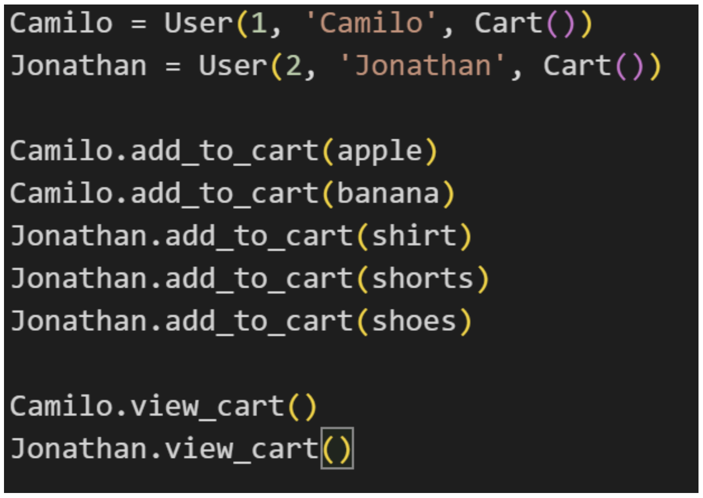
```
apple = DigitalProduct(34, 'Apple', 5, 2, '1MB', 'www.apple.com/download')
banana = DigitalProduct(35, 'Banana', 3, 3, '2MB', 'www.banana.com/download')

shirt = PhysicalProduct(36, 'Shirt', 26, 1, '0.5', 'L', 50)
shorts = PhysicalProduct(37, 'Pants', 15, 2, '1', 'M', 70)
shoes = PhysicalProduct(38, 'Shoes', 90, 1, '1.2', '8', 100)


apple.get_product_info()
banana.get_product_info()
shirt.get_product_info()
shorts.get_product_info()
shoes.get_product_info()
```

--Using get_product_info() method to test and see product information



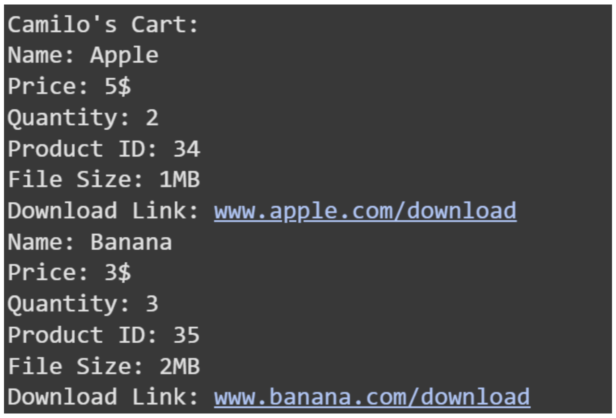
```
Name: Apple
Price: 5$
Quantity: 2
Product ID: 34
File Size: 1MB
Download Link: www.apple.com/download
Name: Banana
Price: 3$
Quantity: 3
Product ID: 35
File Size: 2MB
Download Link: www.banana.com/download
Name: Shirt
Price: 26$
Quantity: 1
Product ID: 36
Weigth: 0.5kg
Dimensions: L
Shiping Cost: 50
Name: Pants
Price: 15$
Quantity: 2
Product ID: 37
Weigth: 1kg
Dimensions: M
Shiping Cost: 70
Name: Shoes
Price: 90$
Quantity: 1
Product ID: 38
Weigth: 1.2kg
Dimensions: 8
Shiping Cost: 100
```

- Create 2 instances of the User class and add the digital products to the user1's cart and physical products to the user2's cart

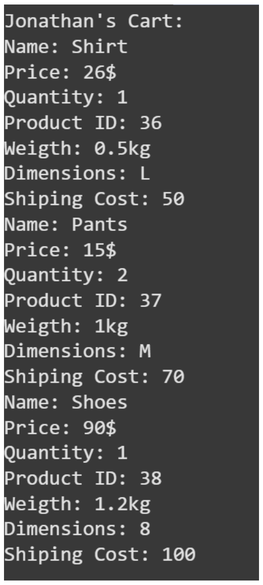


```python
Camilo = User(1, 'Camilo', Cart())
Jonathan = User(2, 'Jonathan', Cart())

Camilo.add_to_cart(apple)
Camilo.add_to_cart(banana)
Jonathan.add_to_cart(shirt)
Jonathan.add_to_cart(shorts)
Jonathan.add_to_cart(shoes)

Camilo.view_cart()
Jonathan.view_cart()
```

- Verify the cart of each user with view_cart() method

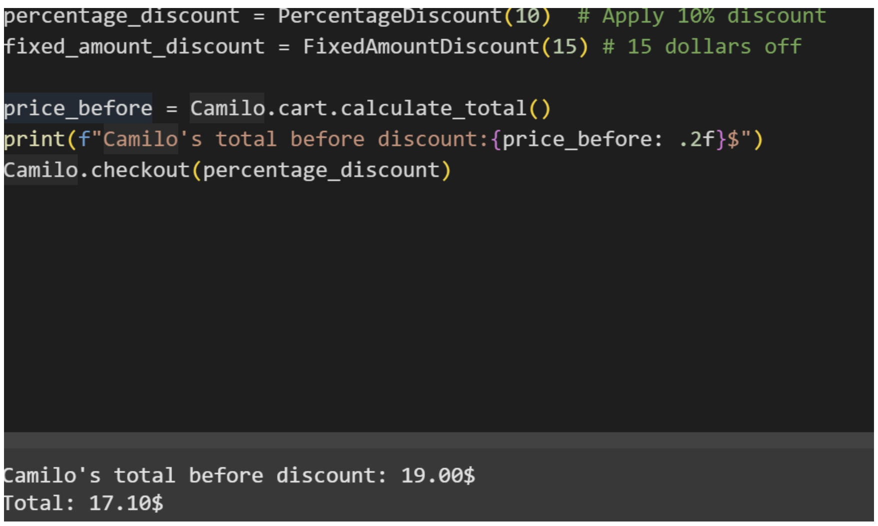


```
Camilo's Cart:
Name: Apple
Price: 5$
Quantity: 2
Product ID: 34
File Size: 1MB
Download Link: www.apple.com/download
Name: Banana
Price: 3$
Quantity: 3
Product ID: 35
File Size: 2MB
Download Link: www.banana.com/download
```



```
Jonathan's Cart:
Name: Shirt
Price: 26$
Quantity: 1
Product ID: 36
Weigth: 0.5kg
Dimensions: L
Shiping Cost: 50
Name: Pants
Price: 15$
Quantity: 2
Product ID: 37
Weigth: 1kg
Dimensions: M
Shiping Cost: 70
Name: Shoes
Price: 90$
Quantity: 1
Product ID: 38
Weigth: 1.2kg
Dimensions: 8
Shiping Cost: 100
```

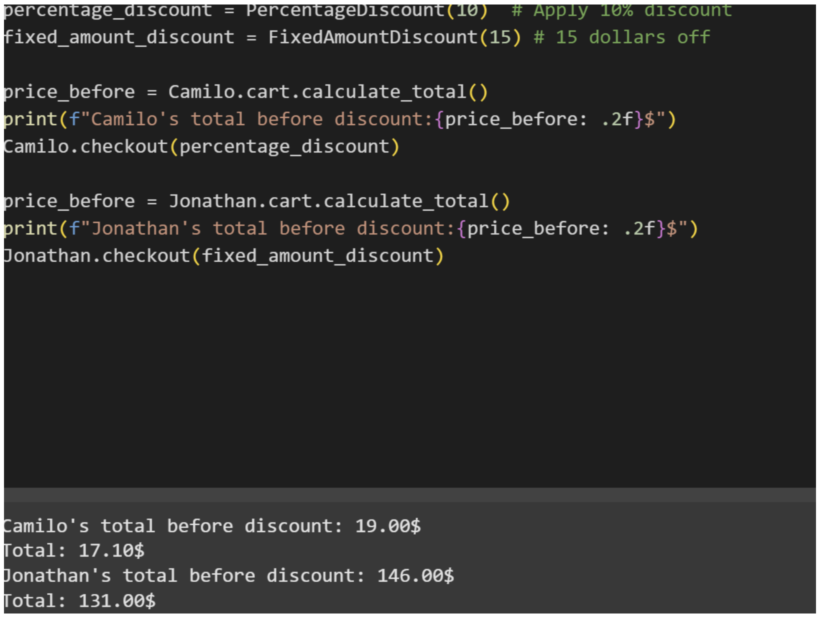
- Create 1 instances of PercentageDiscount and apply it to user1's cart total



```
percentage_discount = PercentageDiscount(10)  # Apply 10% discount
fixed_amount_discount = FixedAmountDiscount(15) # 15 dollars off

price_before = Camilo.cart.calculate_total()
print(f"Camilo's total before discount:{price_before: .2f}$")
Camilo.checkout(percentage_discount)
```

```
Camilo's total before discount: 19.00$
Total: 17.10$
```

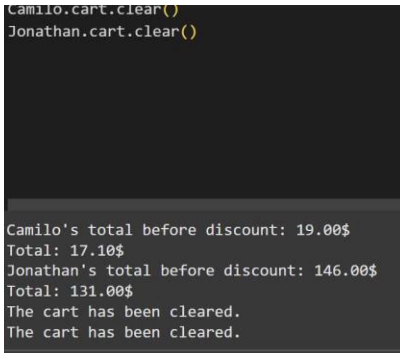
- Create 1 instance of FixedAmountDiscount and apply it to user2's cart total.



```
percentage_discount = PercentageDiscount(10)  # Apply 10% discount
fixed_amount_discount = FixedAmountDiscount(15) # 15 dollars off

price_before = Camilo.cart.calculate_total()
print(f"Camilo's total before discount:{price_before: .2f}$")
Camilo.checkout(percentage_discount)

price_before = Jonathan.cart.calculate_total()
print(f"Jonathan's total before discount:{price_before: .2f}$")
Jonathan.checkout(fixed_amount_discount)
```

```
Camilo's total before discount: 19.00$
Total: 17.10$
Jonathan's total before discount: 146.00$
Total: 131.00$
```

- Perform the checkout process for each user, ensuring the total is calculated correctly, discounts are applied, and the cart is cleared



```
Camilo.cart.clear()
Jonathan.cart.clear()
```

```
Camilo's total before discount: 19.00$
Total: 17.10$
Jonathan's total before discount: 146.00$
Total: 131.00$
The cart has been cleared.
The cart has been cleared.
```

<u>Reflection</u>

In this project I was able to implement principles and understand more about object-oriented programming. Throughout this project I faced different challenges such as applying discount to the user's total, because when I would apply the discount then call the checkout function it would still give me the total of the price before the discount. Which led me to find that I needed to add an if statement in the checkout method to ensure that when a discount is applied it will then apply to the total. Other challenges included user attribute error when for example if I were to try to view a cart (Camilo.view_cart()) then it would give me an error but if I did Camilo.cart.view_cart() then it would work. So, this error had to do with the method being used in the Carts class and not the user's class which I later had add that method into the user's class. Mistakes like that were often in this project dealing with all the other classes and using method from different classes got confusing at times. Some limitations are not having a system where if a product is being added to cart that it will subtract from the total from the e commerce stock. Other limitations are advanced error handling like inputting wrong products or product ids that will give an instant error rather than a built-in error message;