Enhanced E-commerce Shopping Cart Project Report

1. Project Overview

The main purpose of this project was to design and implement a shopping cart system for an e-commerce platform. This system was built using Object-Oriented Programming (OOP) principles by using key concepts such as Encapsulation, Inheritance, Abstraction, and Polymorphism.

The project consists of four main classes:

Product class : Handling different types of products (digital and physical) using the Product class and its derived classes.

Cart System : Managing user carts, adding/removing items, and calculating totals through the Cart class.

User class: Manages user detail and cart interaction

Discount System : Applying discounts to purchases using the Discount class and its subclasses.

2. Project Overview

Product
Attributes - product_id - name - Price - quantity Methods - update_quantity(self,new_quantity) update the quantity to new value - get_product_info(self)   returns product details (id,name,price,quantity)

^

|

|

Digital Product (derived from Product)	Physical Product (derived from Product)
Attributes (get the attributes from product using super()) - file_size - download_link Methods - get_product_info() Overrides method from Product class To include file size and download link .	Attributes (get the attributes from product using super()) - weight - dimension - shipping_cost Methods - get_product_info() Overrides method from Product class To include file size and download link .

Cart
Attributes - __cart_items Methods - add_product(self,product) Adds a product to the cart - remove_product(self,product_id)   removes a product from the cart - view_cart(self)   display all product in the cart - calculate_total() calculate the total cost of all product in cart

User
Attributes - user_id - name - cart (Cart instance) Methods - add_to_cart(self,product) adds a product to the user's cart - remove_from_cart(self,product_id)   removes a product from the user's cart - checkout(self,discount=None)   calculate total, applies a discount and clear the cart

Discount (Parent Abstract class)
Method - apply_disocunt(self,total_amount)   abstract method to apply a discount

PercentageDiscount (child class derived from discount)
Attributes - percentage Method - apply_discount(self, total_amount) applies percentage discount

FixedAmountDiscount (child class derived from discount)
Attributes - amount Method - apply_discount(self, total_amount) applies fixed amount discount

Brief summary on developed classes

1. Product (Parent class) : The products class is a foundation for all product types. It Stores product details. It has a method to update the quantity and another method to print product details.
2. DigitalProduct (child class Inherits from Product): The DigitalProduct class extends Product by adding new attributes specific to digital products like file_size and download_link. It overrides the method to display the extra attributes of this class.
3. PhysicalProduct (child class Inherits from Product): The PhysicalProduct class extends Product by adding attributes like weight, dimension and shipping cost. It overrides the method to display the extra attributes of this class.

4. Cart: The Cart class stores a collection of products for a user. It has multiple methods like adding/ removing items ,calculating the total cost and a method to view the user's cart. It keeps the cart items private.
5. User: The user class uses abstraction to hide the complexity of cart operations from the user. Each user has an id, name and their own cart. It has methods like add/remove product from cart and a checkout method which prints the total after discount applied.
6. Discount (Abstract base class): The discount class defines a common structure for other discount classes. It forces other discount classes to have the apply discount method.
7. PercentageDiscount(Inherits from Discount): Applies a percentage discount and uses an abstract method from the discount class.
8. FixedAmountDiscount(Inherits from Discount): Applies a fixed amount discount and uses an abstract method from the discount class.

3. Instruction for Use

1. Create Products Instances (Digital or Physical product)

2. Create User instances and give a cart to each user

3. Add product to the cart using add_to_cart() method

4.View the user cart using view_cart() method

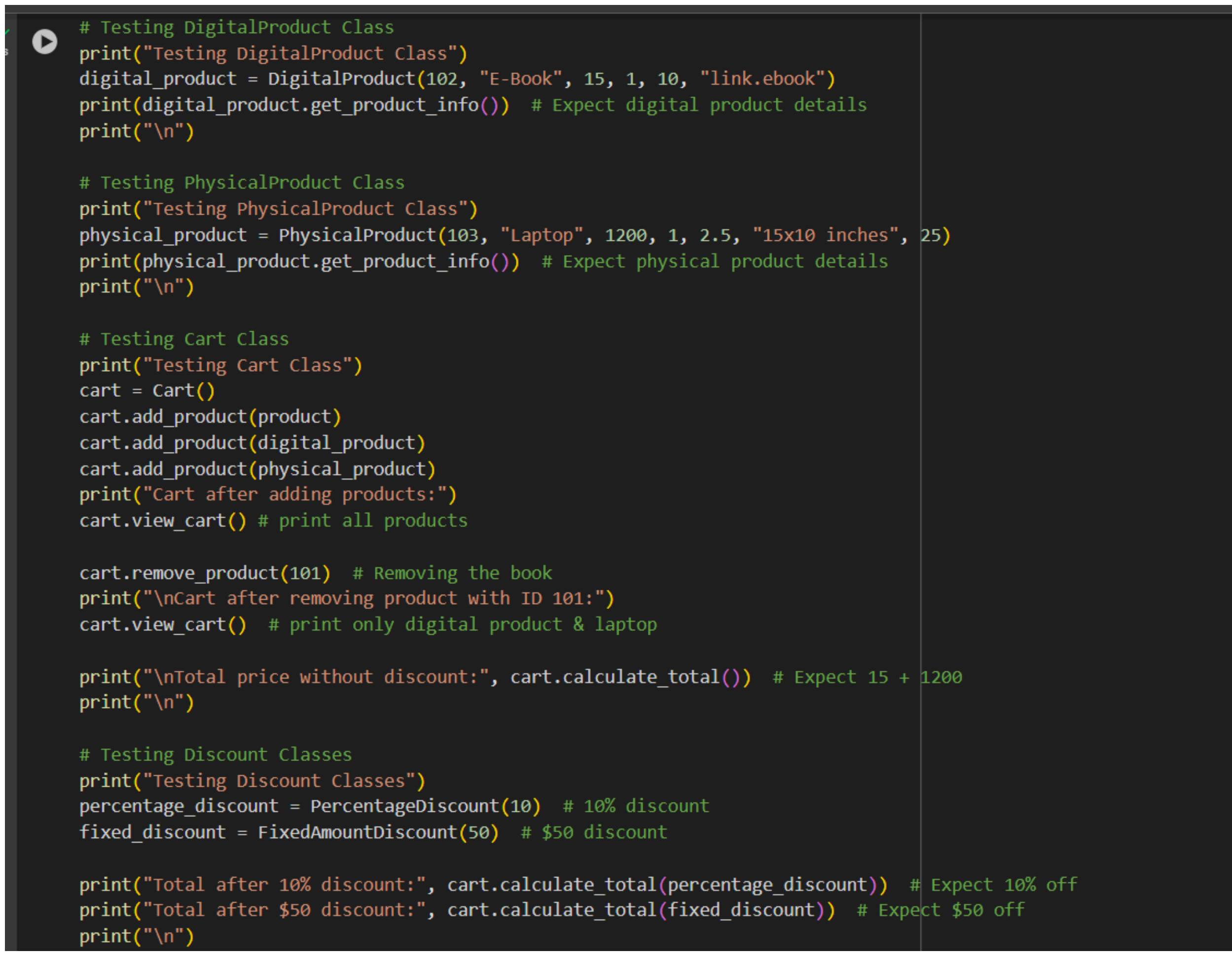
5.Apply discount if the user have any

6.Then do the checkout, which gives the total to the user after discount is applied.

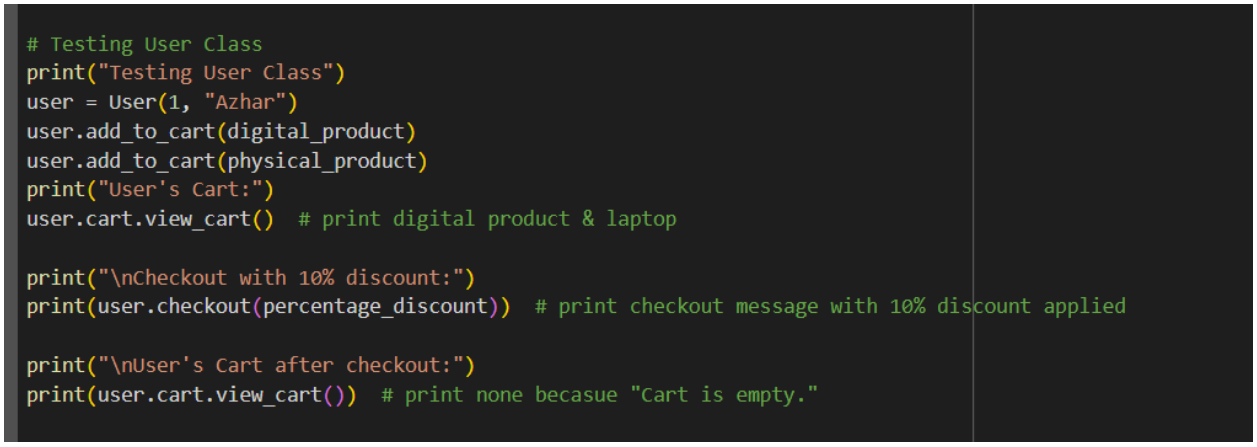
4. Verification Of Code Functionality

- Include screenshots or examples of code execution (Here I took picture of using every method)

Code

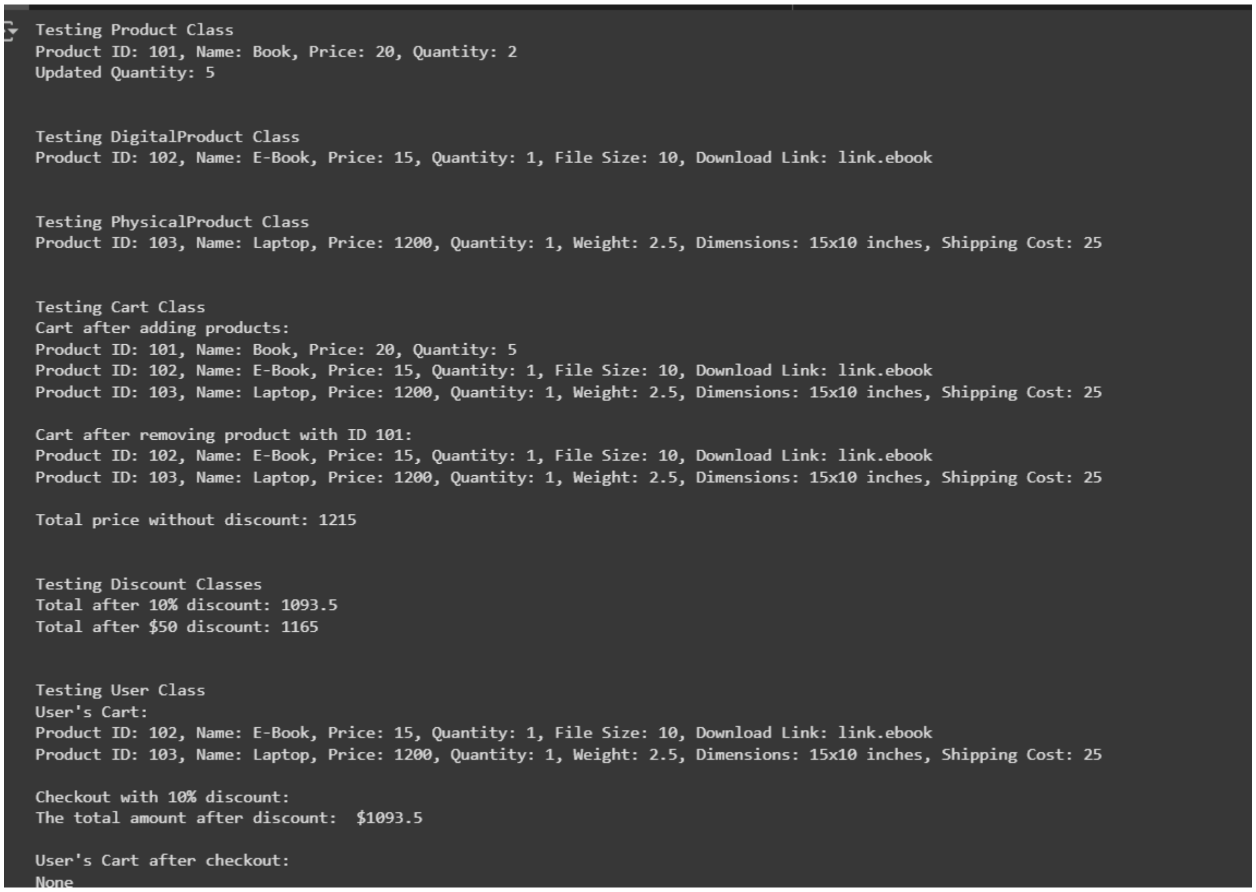


```python
# Testing DigitalProduct Class
print("Testing DigitalProduct Class")
digital_product = DigitalProduct(102, "E-Book", 15, 1, 10, "link.ebook")
print(digital_product.get_product_info())  # Expect digital product details
print("\n")

# Testing PhysicalProduct Class
print("Testing PhysicalProduct Class")
physical_product = PhysicalProduct(103, "Laptop", 1200, 1, 2.5, "15x10 inches", 25)
print(physical_product.get_product_info())  # Expect physical product details
print("\n")

# Testing Cart Class
print("Testing Cart Class")
cart = Cart()
cart.add_product(product)
cart.add_product(digital_product)
cart.add_product(physical_product)
print("Cart after adding products:")
cart.view_cart() # print all products

cart.remove_product(101)  # Removing the book
print("\nCart after removing product with ID 101:")
cart.view_cart()  # print only digital product & laptop

print("\nTotal price without discount:", cart.calculate_total())  # Expect 15 + 1200
print("\n")

# Testing Discount Classes
print("Testing Discount Classes")
percentage_discount = PercentageDiscount(10)  # 10% discount
fixed_discount = FixedAmountDiscount(50)  # $50 discount

print("Total after 10% discount:", cart.calculate_total(percentage_discount))  # Expect 10% off
print("Total after $50 discount:", cart.calculate_total(fixed_discount))  # Expect $50 off
print("\n")
```



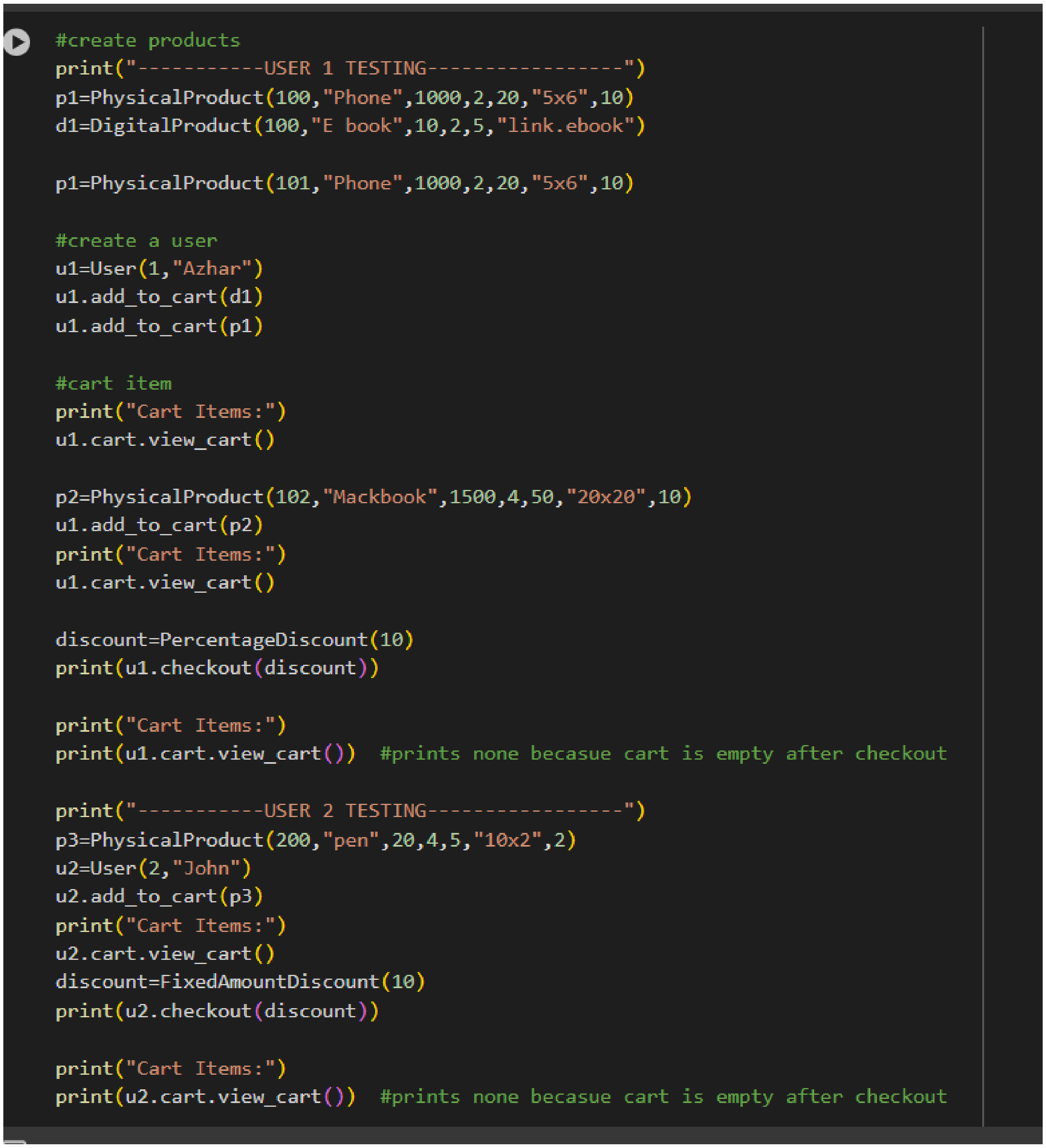
```python
# Testing User Class
print("Testing User Class")
user = User(1, "Azhar")
user.add_to_cart(digital_product)
user.add_to_cart(physical_product)
print("User's Cart:")
user.cart.view_cart()  # print digital product & laptop

print("\nCheckout with 10% discount:")
print(user.checkout(percentage_discount))  # print checkout message with 10% discount applied

print("\nUser's Cart after checkout:")
print(user.cart.view_cart())  # print none becasue "Cart is empty."
```

Result



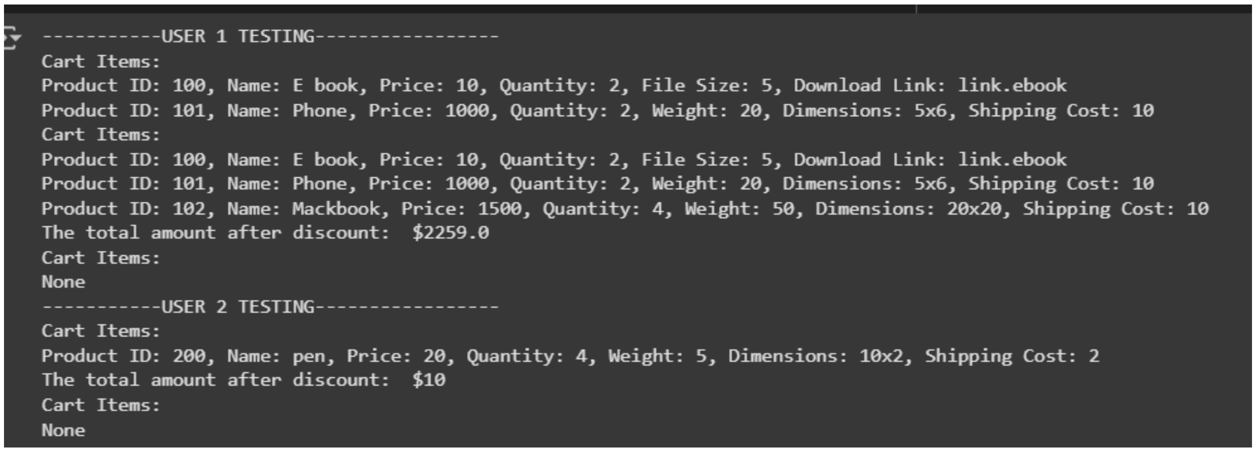
```
Testing Product Class
Product ID: 101, Name: Book, Price: 20, Quantity: 2
Updated Quantity: 5


Testing DigitalProduct Class
Product ID: 102, Name: E-Book, Price: 15, Quantity: 1, File Size: 10, Download Link: link.ebook


Testing PhysicalProduct Class
Product ID: 103, Name: Laptop, Price: 1200, Quantity: 1, Weight: 2.5, Dimensions: 15x10 inches, Shipping Cost: 25


Testing Cart Class
Cart after adding products:
Product ID: 101, Name: Book, Price: 20, Quantity: 5
Product ID: 102, Name: E-Book, Price: 15, Quantity: 1, File Size: 10, Download Link: link.ebook
Product ID: 103, Name: Laptop, Price: 1200, Quantity: 1, Weight: 2.5, Dimensions: 15x10 inches, Shipping Cost: 25

Cart after removing product with ID 101:
Product ID: 102, Name: E-Book, Price: 15, Quantity: 1, File Size: 10, Download Link: link.ebook
Product ID: 103, Name: Laptop, Price: 1200, Quantity: 1, Weight: 2.5, Dimensions: 15x10 inches, Shipping Cost: 25

Total price without discount: 1215


Testing Discount Classes
Total after 10% discount: 1093.5
Total after $50 discount: 1165


Testing User Class
User's Cart:
Product ID: 102, Name: E-Book, Price: 15, Quantity: 1, File Size: 10, Download Link: link.ebook
Product ID: 103, Name: Laptop, Price: 1200, Quantity: 1, Weight: 2.5, Dimensions: 15x10 inches, Shipping Cost: 25

Checkout with 10% discount:
The total amount after discount:  $1093.5

User's Cart after checkout:
None
```

Code execution



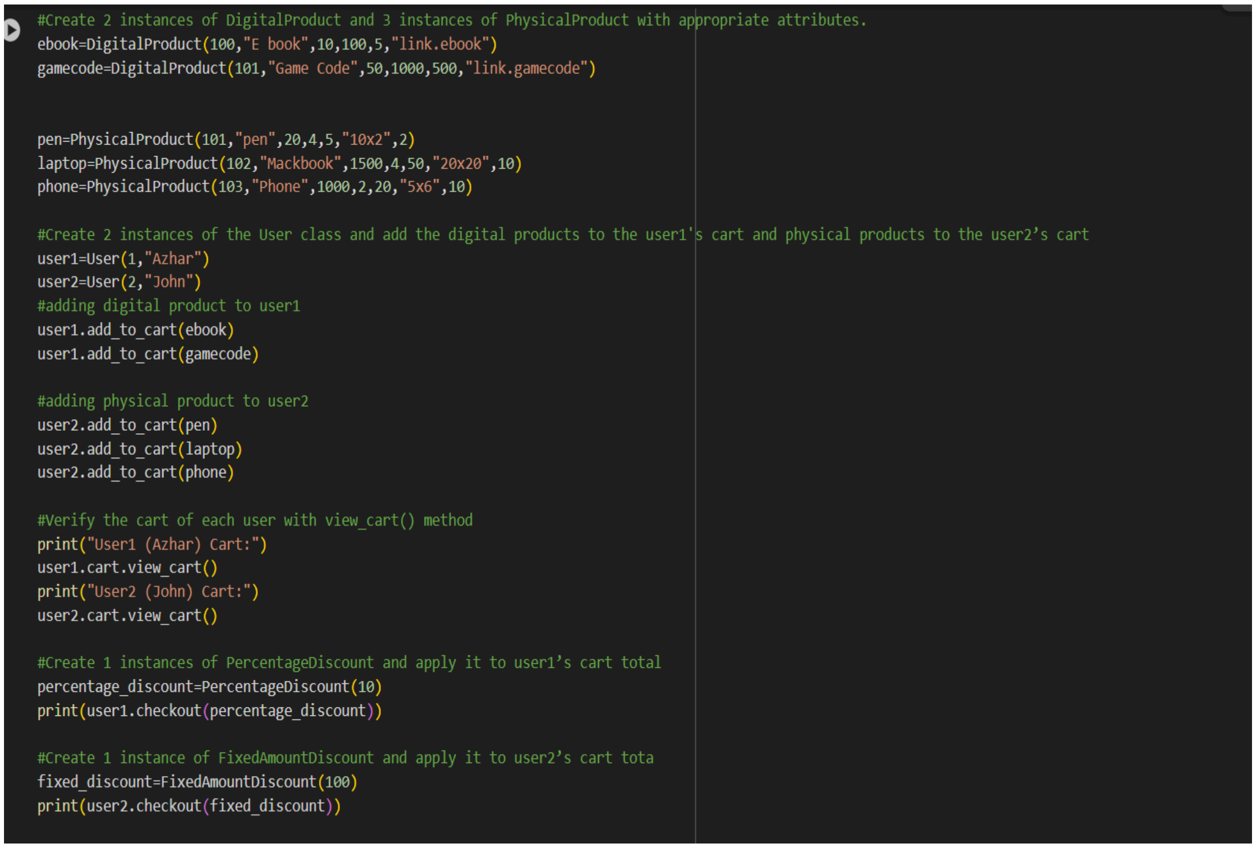
```python
#create products
print("-----------USER 1 TESTING-----------------")
p1=PhysicalProduct(100,"Phone",1000,2,20,"5x6",10)
d1=DigitalProduct(100,"E book",10,2,5,"link.ebook")

p1=PhysicalProduct(101,"Phone",1000,2,20,"5x6",10)

#create a user
u1=User(1,"Azhar")
u1.add_to_cart(d1)
u1.add_to_cart(p1)

#cart item
print("Cart Items:")
u1.cart.view_cart()

p2=PhysicalProduct(102,"Mackbook",1500,4,50,"20x20",10)
u1.add_to_cart(p2)
print("Cart Items:")
u1.cart.view_cart()

discount=PercentageDiscount(10)
print(u1.checkout(discount))

print("Cart Items:")
print(u1.cart.view_cart())  #prints none becasue cart is empty after checkout

print("-----------USER 2 TESTING-----------------")
p3=PhysicalProduct(200,"pen",20,4,5,"10x2",2)
u2=User(2,"John")
u2.add_to_cart(p3)
print("Cart Items:")
u2.cart.view_cart()
discount=FixedAmountDiscount(10)
print(u2.checkout(discount))

print("Cart Items:")
print(u2.cart.view_cart())  #prints none becasue cart is empty after checkout
```

Result

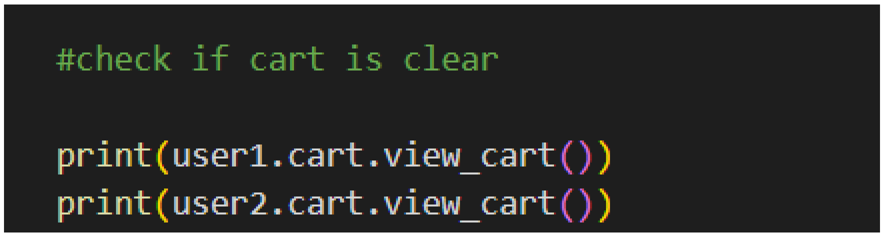


```
-----------USER 1 TESTING-----------------
Cart Items:
Product ID: 100, Name: E book, Price: 10, Quantity: 2, File Size: 5, Download Link: link.ebook
Product ID: 101, Name: Phone, Price: 1000, Quantity: 2, Weight: 20, Dimensions: 5x6, Shipping Cost: 10
Cart Items:
Product ID: 100, Name: E book, Price: 10, Quantity: 2, File Size: 5, Download Link: link.ebook
Product ID: 101, Name: Phone, Price: 1000, Quantity: 2, Weight: 20, Dimensions: 5x6, Shipping Cost: 10
Product ID: 102, Name: Mackbook, Price: 1500, Quantity: 4, Weight: 50, Dimensions: 20x20, Shipping Cost: 10
The total amount after discount:  $2259.0
Cart Items:
None
-----------USER 2 TESTING-----------------
Cart Items:
Product ID: 200, Name: pen, Price: 20, Quantity: 4, Weight: 5, Dimensions: 10x2, Shipping Cost: 2
The total amount after discount:  $10
Cart Items:
None
```

Required Sample Scenario

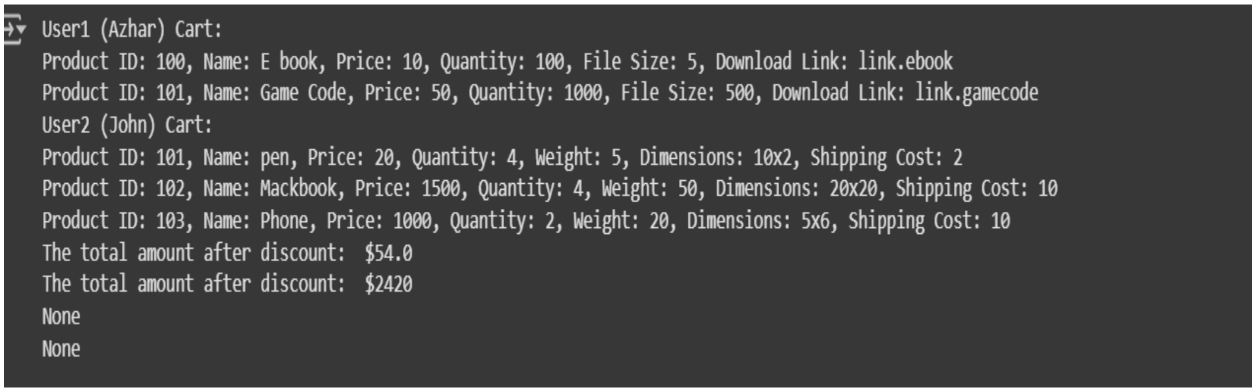


```python
#Create 2 instances of DigitalProduct and 3 instances of PhysicalProduct with appropriate attributes.
ebook=DigitalProduct(100,"E book",10,100,5,"link.ebook")
gamecode=DigitalProduct(101,"Game Code",50,1000,500,"link.gamecode")


pen=PhysicalProduct(101,"pen",20,4,5,"10x2",2)
laptop=PhysicalProduct(102,"Mackbook",1500,4,50,"20x20",10)
phone=PhysicalProduct(103,"Phone",1000,2,20,"5x6",10)

#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
user1=User(1,"Azhar")
user2=User(2,"John")
#adding digital product to user1
user1.add_to_cart(ebook)
user1.add_to_cart(gamecode)

#adding physical product to user2
user2.add_to_cart(pen)
user2.add_to_cart(laptop)
user2.add_to_cart(phone)

#Verify the cart of each user with view_cart() method
print("User1 (Azhar) Cart:")
user1.cart.view_cart()
print("User2 (John) Cart:")
user2.cart.view_cart()

#Create 1 instances of PercentageDiscount and apply it to user1’s cart total
percentage_discount=PercentageDiscount(10)
print(user1.checkout(percentage_discount))

#Create 1 instance of FixedAmountDiscount and apply it to user2’s cart tota
fixed_discount=FixedAmountDiscount(100)
print(user2.checkout(fixed_discount))
```



```
#check if cart is clear

print(user1.cart.view_cart())
print(user2.cart.view_cart())
```

Result



```
User1 (Azhar) Cart:
Product ID: 100, Name: E book, Price: 10, Quantity: 100, File Size: 5, Download Link: link.ebook
Product ID: 101, Name: Game Code, Price: 50, Quantity: 1000, File Size: 500, Download Link: link.gamecode
User2 (John) Cart:
Product ID: 101, Name: pen, Price: 20, Quantity: 4, Weight: 5, Dimensions: 10x2, Shipping Cost: 2
Product ID: 102, Name: Mackbook, Price: 1500, Quantity: 4, Weight: 50, Dimensions: 20x20, Shipping Cost: 10
Product ID: 103, Name: Phone, Price: 1000, Quantity: 2, Weight: 20, Dimensions: 5x6, Shipping Cost: 10
The total amount after discount:  $54.0
The total amount after discount:  $2420
None
None
```

5.Discussion and Conclusion

Your finding through this project

1. This project showed how effective OOP is in these types of projects . OOP helps users for code reusability, easier maintenance and makes everything organized. This project made me realize why almost every big company utilizes OOP for their big projects. Through this project, I realized the importance of encapsulation, especially in keeping the cart private. This shows how we have to use a public method to access the private attribute of the user's cart. This makes sure the user's data is secured. Implementing inheritance and polymorphism made the system more efficient and it also saves lots of

time as I could reuse the code of the parent class. Also, using Abstraction in Discount class showed me how it serves as a blueprint for other classes and it forces those classes to implement the method of the parent abstract class. Overall this project improved my understanding of OOP concepts and their real world applications.

Challenges Faced

1. While working on this project I faced many challenges like learning abstraction through YouTube tutorials and then applying that knowledge to the project. Another challenge was planning before coding, which initially made my code unorganized and wasted time. As a result, I had to put extra effort to structure the code properly.

Area for improvement

1. One area of improvement would be if a user tries to add more products than the available stock, the system should give a better message. We can maybe use “try/Catch” for this. Another improvement is that After a user buys the product , the product quantity should go down by itself. Finally, another improvement is to add user login functionality which would allow the system to store user data.