James Macartney

Project Report

Description:

This project is a simple shopping cart system designed for an e-commerce platform using object-oriented programming. It supports both digital and physical products, allows discounts to be applied, and manages the checkout process. The system is built around different product types and discount options, such as percentage-based and fixed amount discounts.

Overview:

The project is organized into several modular classes:

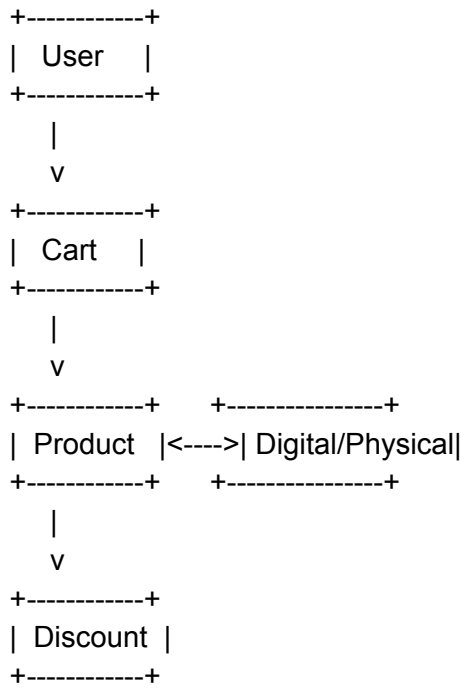
- ***Product**: Serves as the base class with key attributes like price and quantity.*
- ***DigitalProduct & PhysicalProduct**: Extend the base class, handling product-specific details.*
- ***Cart**: Keeps track of products, calculates totals, and applies discounts.*
- ***User**: Represents a shopper, allowing them to manage their cart and proceed to checkout.*
- ***Discount**: An abstract class with specific implementations like PercentageDiscount and FixedAmountDiscount.*

Structure

The code is broken into modular classes:

- ***Product****: Base class for products with attributes like price and quantity.*
- ***DigitalProduct*** *and* ***PhysicalProduct****: Specializations of* `Product` *for digital and physical products.*
- ***Cart****: Manages the cart's content, calculates totals, and applies discounts.*
- ***User****: Simplifies cart operations for users, letting them add/remove products and checkout.*
- ***Discount****: Abstract class for discounts with* `PercentageDiscount` *and* `FixedAmountDiscount` *to apply discounts to the cart.*

Simple Flowchart



Instructions:

1. ***Creating Products****: Define digital or physical products with relevant details like price, quantity, and size.*
2. ***Adding to Cart****: Users can add products to their cart using an* `add_to_cart()` *method.*
3. ***Applying Discounts****: Discounts are applied by creating instances of either* `PercentageDiscount` *or* `FixedAmountDiscount`*.*
4. ***Checkout Process****: The* `checkout()` *method calculates the final amount, applies any discounts, and clears the cart.*

Testing Example

1. *Create two digital and three physical product instances.*
2. *Have two users: one adding digital products to their cart, the other adding physical ones.*
3. *Apply a 10% discount to the first user's cart and a $50 discount to the second user's.*
4. *Complete the checkout process for both users and display the final total for each.*

```python
# Test the classes

# Creating products
digital_product = DigitalProduct(1, "E-book", 15.0, 1, 5, "http://downloadlink.com")
physical_product = PhysicalProduct(2, "Laptop", 1000.0, 2, 2.5, "15x10x1 inches", 20.0)

# Creating a user and adding products to the cart
user = User(1, "John Doe")
user.add_to_cart(digital_product)
user.add_to_cart(physical_product)

# Viewing the cart
print("Cart Contents:")
for product_info in user.cart.view_cart():
    print(product_info)

# Applying discount
percentage_discount = PercentageDiscount(10)  # 10% discount
total_with_discount = user.checkout(percentage_discount)
print(f"Total after discount: {total_with_discount}")

# Re-adding products for fixed amount discount test
user.add_to_cart(digital_product)
user.add_to_cart(physical_product)
fixed_discount = FixedAmountDiscount(50)  # Fixed $50 discount
total_with_fixed_discount = user.checkout(fixed_discount)
print(f"Total after fixed discount: {total_with_fixed_discount}")

# Create products
digital1 = DigitalProduct(1, "E-book", 15.0, 1, 5, "http://download.com")
digital2 = DigitalProduct(2, "Software", 50.0, 2, 50, "http://software.com")
physical1 = PhysicalProduct(3, "Laptop", 1000.0, 2, 2.5, "15x10x1", 20.0)
physical2 = PhysicalProduct(4, "Phone", 500.0, 1, 0.3, "6x3x0.5", 10.0)
physical3 = PhysicalProduct(5, "Tablet", 300.0, 3, 0.5, "8x5x0.5", 15.0)
```

```python
# Create users and add products to their carts
user1 = User(1, "John")
user2 = User(2, "Jane")
user1.add_to_cart(digital1)
user1.add_to_cart(digital2)
user2.add_to_cart(physical1)
user2.add_to_cart(physical2)
user2.add_to_cart(physical3)

# Apply discounts and checkout
percentage_discount = PercentageDiscount(10)
fixed_discount = FixedAmountDiscount(50)

user1_total = user1.checkout(percentage_discount)
user2_total = user2.checkout(fixed_discount)

print(f"User 1 Total: {user1_total}")
print(f"User 2 Total: {user2_total}")
```

Shell

```
>>> %Run inventoryjjm.py

  Cart Contents:
  ID: 1, Name: E-book, Price: 15.0, Quantity: 1, File Size: 5MB, Download Link: http://downloadlink.com
  ID: 2, Name: Laptop, Price: 1000.0, Quantity: 2, Weight: 2.5kg, Dimensions: 15x10x1 inches, Shipping Cost: 20.0
  Total after discount: 1813.5
  Total after fixed discount: 1965.0
  User 1 Total: 103.5
  User 2 Total: 3350.0
```

Conclusion:

This project demonstrates how to build a simple, object-oriented shopping cart system. By organizing the system into modular classes, it is easy to manage products, carts, and discounts. Users can interact with the system simply and effectively.

***Challenges**: Managing different product types and discounts required careful design to ensure correct calculations.*

***Limitations**: The system can be extended with more advanced features, such as quantity updates and more complex discount structures.*