Republic of the Philippines
SURIGAO DEL NORTE STATE UNIVERSITY
Narciso Street, Surigao City 8400, Philippines
"For Nation's Greater"

In Partial Fulfillment of the Requirements for the

CS 223 - Object-Oriented Programming

"FOUR PRINCIPLES OF OBJECT ORIENTED PROGRAMMING"

Presented to:

Dr. Unife O. Cagas
Professor

Prepared by:

Confesor, Mailyn L.
BSCS 2A2 Student

"E-commerce Platform"

Project Title

Project Description

The software application known as the e-commerce platform was created to make online shopping and sales easier. It offers an easily navigable platform for customers to view items, place them in their carts, and finalize purchases with safety. The platform provides tools for merchants to organize their product listings, fulfill orders, and create analytics.

Objectives

1. Offer customers a broad selection of products, safe payment methods, and fast order fulfillment for a smooth shopping experience.
2. Allow retailers to efficiently oversee their e-commerce shops, which includes managing product postings, controlling inventory, and completing orders.
3. Guarantee data security and privacy for customers and merchants through the implementation of strong security measures and compliance with industry standards.
4. Simplify the process of connecting with different payment gateways, shipping providers, and other third-party services in order to improve the functionality of the platform.

Importance and Contribution of the Project

The e-commerce platform is essential in the digital economy as it allows companies to reach a larger audience and grow their sales channels. It helps boost the expansion of online business by offering a trustworthy and effective platform for carrying out transactions. The platform also brings about job opportunities in different fields, including web development, digital marketing, and customer support.

Four Principles of Object-Oriented Programming with Code

Abstraction

The `Product` abstract class defines the common properties and methods for products in the e-commerce platform, providing a blueprint for concrete product types.

```python
# Abstraction
from abc import ABC, abstractmethod

class Product(ABC):
    def __init__(self, name, price):
        self.name = name
        self.price = price

    @abstractmethod
    def display_details(self):
        pass
```

Inheritance

The `Electronics` class inherits from the `Product` class, allowing it to inherit the common properties and methods while adding specific attributes and behaviors for electronic products.

```
# Inheritance
class Electronics(Product):
    def __init__(self, name, price, brand, model):
        super().__init__(name, price)
        self.brand = brand
        self.model = model

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Price: ${self.price}")
        print(f"Brand: {self.brand}")
        print(f"Model: {self.model}")

class Clothing(Product):
    def __init__(self, name, price, size, color):
        super().__init__(name, price)
        self.size = size
        self.color = color

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Price: ${self.price}")
        print(f"Size: {self.size}")
        print(f"Color: {self.color}")
```

Encapsulation

The `ShoppingCart` class encapsulates the list of items in the shopping cart using private attributes and provides methods to manage and display the cart contents, ensuring data integrity and security.

```
# Encapsulation
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_to_cart(self, product):
        self.items.append(product)

    def remove_from_cart(self, product):
        self.items.remove(product)

    def display_cart(self):
        print("Shopping Cart:")
        for product in self.items:
            product.display_details()
            print()
```

Polymorphism

In the `main` block, an instance of the `Electronics` class is treated as a `Product` object, allowing it to be added to the shopping cart and displayed using the common `display_details` method.

```
55 # Polymorphism
56 if __name__ == "__main__":
57     electronics = Electronics("Smartphone", 599.99, "Apple", "iPhone 13")
58     clothing = Clothing("T-Shirt", 29.99, "M", "Blue")
59
60     cart = ShoppingCart()
61     cart.add_to_cart(electronics)
62     cart.add_to_cart(clothing)
63
64     cart.display_cart()
```

Hardware and Software Used

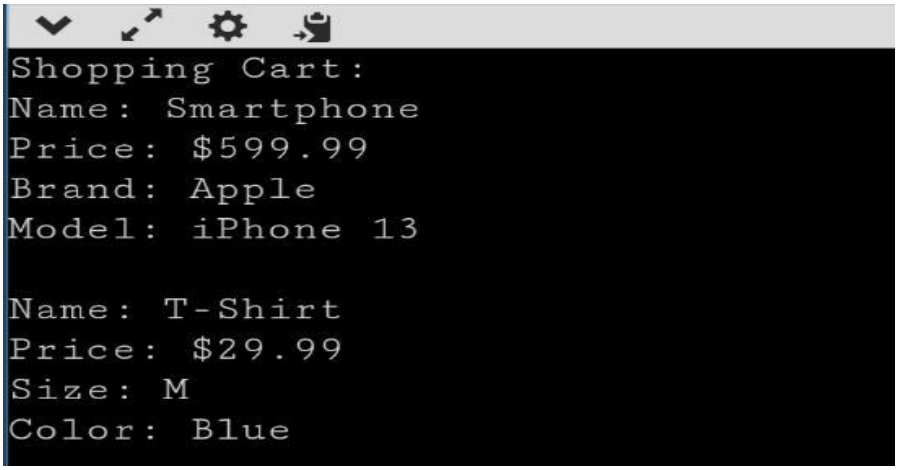
SOFTWARE USED:

- Online GDB

HARDWARE USED:

- Mobile Phone
- School PC

Output



The program output shows the items in the shopping cart, beginning with a heading labeled "Shopping Cart:". Afterward, the specifics of the products included are displayed: the first item is a smartphone named "Smartphone" with a price tag of $599.99, made by brand "Apple", and model "iPhone 13", capturing the characteristics of the `Electronics` category. The second item, a t-shirt, is presented with the designation "T-Shirt", costing $29.99, being size "M", and color "Blue", presenting the attributes specified in the `Clothing` category. The result shows the organized display of product information in the cart, showcasing the various details included in each product category and the versatile functionality facilitated by the OOP architecture.

Code

```python
# Abstraction
from abc import ABC, abstractmethod

class Product(ABC):
    def __init__(self, name, price):
        self.name = name
        self.price = price

    @abstractmethod
    def display_details(self):
        pass

# Inheritance
class Electronics(Product):
    def __init__(self, name, price, brand, model):
        super().__init__(name, price)
        self.brand = brand
        self.model = model

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Price: ${self.price}")
        print(f"Brand: {self.brand}")
        print(f"Model: {self.model}")

class Clothing(Product):
    def __init__(self, name, price, size, color):
        super().__init__(name, price)
        self.size = size
        self.color = color

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Price: ${self.price}")
        print(f"Size: {self.size}")
        print(f"Color: {self.color}")

# Encapsulation
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_to_cart(self, product):
        self.items.append(product)

    def remove_from_cart(self, product):
        self.items.remove(product)

    def display_cart(self):
        print("Shopping Cart:")
        for product in self.items:
            product.display_details()
            print()

# Polymorphism
if __name__ == "__main__":
    electronics = Electronics("Smartphone", 599.99, "Apple", "iPhone 13")
    clothing = Clothing("T-Shirt", 29.99, "M", "Blue")

    cart = ShoppingCart()
    cart.add_to_cart(electronics)
    cart.add_to_cart(clothing)

    cart.display_cart()

```

User Guide

Product Class

- `Product` serves as the foundational class for all products.
- It contains properties for both `name` and `price`.
- It includes a method `display_details()` that subclasses are required to implement.

Electronics Class

- `Product` is the superclass of `Electronics` and includes items that are electronic.
- It contains extra features related to `brand` and `model`.
- It utilizes the `display_details()` function to showcase all the characteristics of an electronic item.

Clothing Class

- `Clothing` falls under the category of `Product` and is used to refer to clothing items.
- It includes extra characteristics related to `size` and `color`.
- It includes the `display_details()` function to show all the characteristics of a clothing item.

Class for managing shopping carts.

- `ShoppingCart` shows a group of items that can be included, deleted, and shown.
- There is a function called `add_to_cart(product)` which is used to include a product in the cart.
- It includes a function `remove_from_cart(product)` for taking out a product from the cart.
- It includes a function called `display_cart()` that shows all the items in the cart.

Primary Role

- The primary objective is to generate objects of 'Electronics' and 'Clothing', place them in a 'ShoppingCart', and showcase what is inside the cart.

Individuals can utilize this manual to grasp the utilization and interaction with the classes and functions available in the software.

REFERENCES

- https://www.onlinegdb.com/
- https://www.w3schools.com/
- https://python.org/
- https://chatgpt.com/