Ex. No: 7	UIT2201 — Programming and Data Structures
20-05-2023	

Aim:

To execute the following programs and note the output.

PART – A

1. Write a program to model a real-time online shopping system using inheritance. The base class should be called Product, and it should have attributes for the name, price, and quantity of the product. The derived classes should be ElectronicProduct and ClothingProduct, which inherit from Product. Each derived class should have additional attributes specific to that type of product, such as the brand and model for ElectronicProduct, and the size and color for ClothingProduct. Implement methods in each class to display the product information. Additionally, override the display_information() method in the derived classes to include the specific attributes of each product type. Also, implement a function in the derived classes to calculate the total price based on the quantity of the product. Finally, overload the '+' operator in the derived classes to allow adding two products together offering a combo pack with the summed-up price tag.

Code:

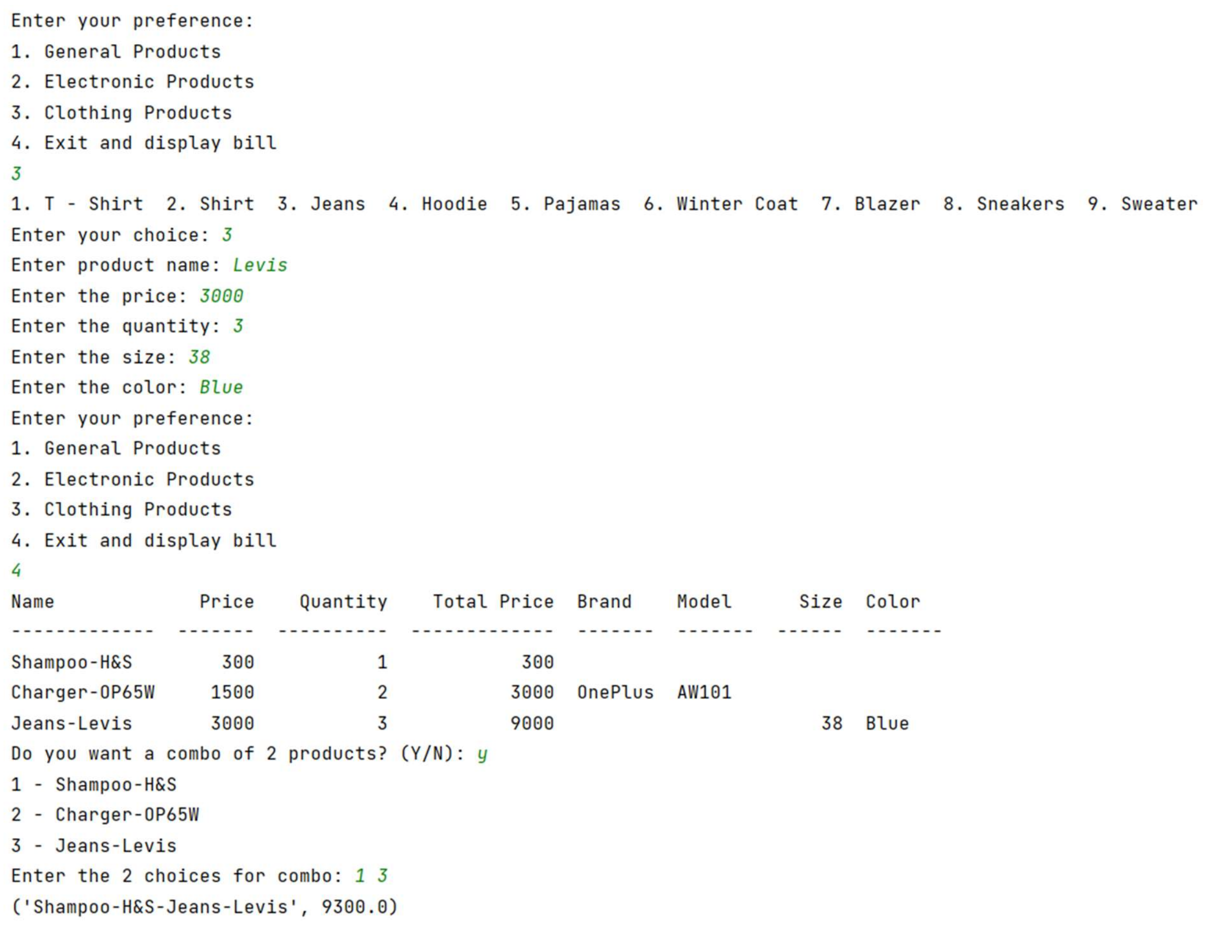
```python
"""
This module implements the concept of inheritance using classes and objects
in python

Original Author: Pranesh Kumar

Created on: 17 May 2023

"""

# importing module for displaying the bill
import tabulate


class Product:
    def __init__(self, name: str = "", price: float = 0.0, quantity: float
= 0.0):
        """
        Constructor of Product class
        Args:
            name (str): name of the product
            price (float): price of the product
            quantity (float): quantity of the product
        """
        self.name = name
        self.price = price
        self.quantity = quantity
        self.items_dict = {}

    def display_information(self):
        """
        Displays the information of the product
        Returns:
            None
        """
        print(f"Name: {self.name} "
              f"Price: {self.price} "
              f"Quantity: {self.quantity} ")

    def get_name(self):
        """
        Returns the name of the product
        Returns:
            name(str): Name of the product
        """
        return self.name

    def get_price(self):
        """
        Returns the price of the product
        Returns:
            price(float): Price of the product
        """
        return self.price

    def get_quantity(self):
        """
        Returns the quantity of the product
        Returns:
            quantity(float): Quantity of the product
        """
        return self.quantity

    def calculate_total_price(self):
        """
        Calculates the total price of the product by multiplying the price
and the quantity
        Returns:
            price(float): total price of the product
        """
        return self.price * self.quantity

    def add_items(self, other):
        if isinstance(other, ElectronicProduct):
            self.items_dict.update({other.get_name(): {"price":
other.get_price(),
                                                        "quantity":
other.get_quantity(),
                                                        "total-price":
other.calculate_total_price(),
                                                        "brand":
other.get_brand(),
                                                        "model":
other.get_model()}})
        elif isinstance(other, ClothingProduct):
            self.items_dict.update({other.get_name(): {"price":
other.get_price(),
                                                        "quantity":
other.get_quantity(),
                                                        "total-price":
other.calculate_total_price(),
                                                                       "size":
other.get_size(),
                                                                       "color":
other.get_color()}})
        elif isinstance(other, Product):
            self.items_dict.update({other.get_name(): {"price":
other.get_price(),
                                                                       "quantity":
other.get_quantity(),
                                                                       "total-price":
other.calculate_total_price()}})

    def get_items(self):
        return self.items_dict

    def __add__(self, other):
        return self.get_name() + "-" + other.get_name(),
self.calculate_total_price() + other.calculate_total_price()


class ElectronicProduct(Product):
    def __init__(self, name: str, price: float, quantity: float, brand:
str, model: str):
        """
        Constructor of ElectronicProduct Class
        Args:
            name  (str): name of the product
            price  (float): price of the product
            quantity  (float): quantity of the product
            brand(str): brand of the product
            model(str): model of the product
        """
        super().__init__(name, price, quantity)
        self.brand = brand
        self.model = model

    def display_information(self):
        """
        Displays the information of the product
        Returns:
            None
        """
        print(f"Name: {self.name} "
              f"Price: {self.price} "
              f"Quantity: {self.quantity} "
              f"Brand: {self.brand} "
              f"Model: {self.model} ")

    def calculate_total_price(self):
        """
        Calculates the total price of the product by multiplying the price
and the quantity
        Returns:
            price(float): total price of the product
        """
        return self.price * self.quantity

    def get_name(self):
        """
        Returns the name of the product
        Returns:
            name(str): Name of the product
        """
        return self.name

    def get_model(self):
        """
        Returns the model of the product
        Returns:
            model(str): Model of the product
        """
        return self.model

    def get_brand(self):
        """
        Returns the brand of the product
        Returns:
            brand(str): Brand of the product
        """
        return self.brand

    def __add__(self, other):
        return self.get_name() + "-" + other.get_name(),
self.calculate_total_price() + other.calculate_total_price()


class ClothingProduct(Product):
    def __init__(self, name: str, price: float, quantity: float, size: str,
color: str):
        """
        Constructor of ElectronicProduct Class
        Args:
            name (str): name of the product
            price (float): price of the product
            quantity (float): quantity of the product
            size(str): size of the product
            color(str): color of the product
        """
        super().__init__(name, price, quantity)
        self.size = size
        self.color = color

    def display_information(self):
        """
        Displays the information of the product
        Returns:
            None
        """
        print(f"Name: {self.name} "
              f"Price: {self.price} "
              f"Quantity: {self.quantity} "
              f"Size: {self.size}"
              f"Color: {self.color}")

    def calculate_total_price(self):
        """
        Calculates the total price of the product by multiplying the price
and the quantity
        Returns:
            price(float): total price of the product
        """
        return self.price * self.quantity
```

```python
    def get_name(self):
        """
        Returns the name of the product
        Returns:
            name(str): Name of the product
        """
        return self.name

    def get_size(self):
        """
        Returns the size of the product
        Returns:
            size(int): size of the product
        """
        return self.size

    def get_color(self):
        """
        Returns the color of the product
        Returns:
            color(str): color of the product
        """
        return self.color

    def __add__(self, other):
        return self.get_name() + "-" + other.get_name(),
self.calculate_total_price() + other.calculate_total_price()


# driver code
if __name__ == "__main__":
    products = [
        "1. Bread",
        "2. Milk",
        "3. Eggs",
        "4. Toothpaste",
        "5. Shampoo",
        "6. Powder",
        "7. Soap",
        "8. Toilet Paper",
        "9. Tissues"
    ]

    electronic_products = [
        "1. Earphones",
        "2. Headphones",
        "3. Pen drive",
        "4. Memory Cards",
        "5. Charger",
        "6. Batteries",
        "7. Mouse",
        "8. Keyboard",
        "9. Laptop",
        "10. Mobile"
    ]

    clothing_products = [
        "1. T - Shirt",
        "2. Shirt",
        "3. Jeans",
        "4. Hoodie",
        "5. Pajamas",
        "6. Winter Coat",
        "7. Blazer",
        "8. Sneakers",
        "9. Sweater",
    ]

    pr = Product()
    my_items = []
    while True:
        print("Enter your preference:"
              "\n1. General Products"
              "\n2. Electronic Products"
              "\n3. Clothing Products"
              "\n4. Exit and display bill")
        ch = input()

        if ch == "1":
            print(*products, sep="  ")
            prod_ch = int(input("Enter your choice: "))
            prod_name = input("Enter product name: ")
            cost = float(input("Enter the price: "))
            qty = float(input("Enter the quantity: "))
            prod_obj = Product(products[prod_ch - 1][3:] + "-" + prod_name,
cost, qty)
            my_items.append(prod_obj)
            pr.add_items(prod_obj)
        elif ch == "2":
            print(*electronic_products, sep="  ")
            prod_ch = int(input("Enter your choice: "))
            prod_name = input("Enter product name: ")
            cost = float(input("Enter the price: "))
            qty = float(input("Enter the quantity: "))
            my_brand = input("Enter the brand: ")
            my_model = input("Enter the model: ")
            prod_obj = ElectronicProduct(electronic_products[prod_ch -
1][3:] + "-" + prod_name, cost, qty, my_brand,
                                             my_model)
            my_items.append(prod_obj)
            pr.add_items(prod_obj)

        elif ch == "3":
            print(*clothing_products, sep="  ")
            prod_ch = int(input("Enter your choice: "))
            prod_name = input("Enter product name: ")
            cost = float(input("Enter the price: "))
            qty = float(input("Enter the quantity: "))
            my_size = input("Enter the size: ")
            my_color = input("Enter the color: ")
            prod_obj = ClothingProduct(clothing_products[prod_ch - 1][3:] +
"-" + prod_name, cost, qty, my_size,
                                           my_color)
            my_items.append(prod_obj)
            pr.add_items(prod_obj)
        elif ch == "4":
            break
        else:
            print("Try again!")
            continue
```

```python
    items_dict = pr.get_items()

    header = ["Name", "Price", "Quantity", "Total Price", "Brand", "Model",
"Size", "Color"]

    values = []

    for key, value in items_dict.items():
        temp = [key]
        temp.extend(value.values())
        if key.split("-")[0] in str(clothing_products):
            temp.extend(value.values())
            temp.insert(4, "")
            temp.insert(5, "")
        values.append(temp)
    print(tabulate.tabulate(values, header))

    combo_ch = input("Do you want a combo of 2 products? (Y/N): ")
    if combo_ch.lower() == "y":
        for idx, item in enumerate(my_items):
            print(idx+1, "-", item.get_name())
        combo_prods = list(map(int, input("Enter the 2 choices for combo:
").split(" ")))
        print(my_items[combo_prods[0]-1] + my_items[combo_prods[1] - 1])
```

Inputs and Output:

```
Enter your preference:
1. General Products
2. Electronic Products
3. Clothing Products
4. Exit and display bill
1
1. Bread  2. Milk  3. Eggs  4. Toothpaste  5. Shampoo  6. Powder  7. Soap  8. Toilet Paper  9. Tissues
Enter your choice: 5
Enter product name: H&S
Enter the price: 300
Enter the quantity: 1
Enter your preference:
1. General Products
2. Electronic Products
3. Clothing Products
4. Exit and display bill
2
1. Earphones  2. Headphones  3. Pen drive  4. Memory Cards  5. Charger  6. Batteries  7. Mouse  8. Keyboard  9. Laptop  10. Mobile
Enter your choice: 5
Enter product name: OP65W
Enter the price: 1500
Enter the quantity: 2
Enter the brand: OnePlus
Enter the model: AW101
```



```
Enter your preference:
1. General Products
2. Electronic Products
3. Clothing Products
4. Exit and display bill
3
1. T - Shirt  2. Shirt  3. Jeans  4. Hoodie  5. Pajamas  6. Winter Coat  7. Blazer  8. Sneakers  9. Sweater
Enter your choice: 3
Enter product name: Levis
Enter the price: 3000
Enter the quantity: 3
Enter the size: 38
Enter the color: Blue
Enter your preference:
1. General Products
2. Electronic Products
3. Clothing Products
4. Exit and display bill
4
Name             Price    Quantity    Total Price  Brand    Model      Size  Color
-------------  -------  ----------  -------------  -------  -------  ------  -------
Shampoo-H&S        300           1            300
Charger-OP65W     1500           2           3000  OnePlus  AW101
Jeans-Levis       3000           3           9000                        38  Blue
Do you want a combo of 2 products? (Y/N): y
1 - Shampoo-H&S
2 - Charger-OP65W
3 - Jeans-Levis
Enter the 2 choices for combo: 1 3
('Shampoo-H&S-Jeans-Levis', 9300.0)
```

========================