**SSN College of Engineering**

Department of Information Technology

UIT2201 — Programming and Data Structures

2022 – 2023

**Exercise — 07**

U. Pranaav | IT-B | 3122225002093

I. AIM:

To 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. Also 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.

II. CODE:

# -\*- coding: utf-8 -\*-

"""

This module provides a series of classes that derive certain

functionalities from a parent class and overload some parent

class operations for each child class. This is a part of the exercises

given under the course UIT2201 (Programming and Data Structures).

In this source code I have executed my own logic. The code

follows good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 17 2023

Revised on Wed May 17 2023

Original Author: U. Pranaav <pranaav2210205@ssn.edu.in>

"""

*class* product:

    '''

    The Product class represents a product with its name, price,

    and quantity. It provides methods for performing various

    operations related to the product.

    The input data is not modified in any way and there are

    no side effects.

    Methods:

        \_\_init\_\_(self, name, price, quantity): Initializes a new instance

        of the Product class with the specified name, price, and quantity.

        \_\_add\_\_(self, other): Adds the quantities and prices of two products

        and displays a combo offer message.

        display\_information(self): Displays the name, price, and quantity

        of the product.

        cost\_calc(self): Calculates and displays the total cost of the

        product.

    '''

*def* \_\_init\_\_(*self*,*name*,*price*,*quantity*):

*self*.name = *name*

*self*.price = *price*

*self*.quantity = *quantity*

*def* \_\_add\_\_(*self*,*other*):

        prods = [*self*.name, *other*.name]

        total\_cost = (*self*.price \* *self*.quantity) + (*other*.price \* *other*.quantity)

        print(*f*"Combo offer for {*self*.name} and {*other*.name} !\n For a price of {total\_cost}")

*def* display\_information(*self*):

        print("Name is : ",*self*.name)

        print("Price is : ",*self*.price)

        print("Quantity is : ",*self*.quantity)

*def* cost\_calc(*self*):

        print("Total cost is : ", *self*.price \* *self*.quantity)

*class* ElectronicProduct(product):

    '''

    The ElectronicProduct class represents an electronic product,

    which is a specialized type of Product. It inherits the attributes

    and methods from the Product class and adds additional attributes

    for brand and model.

    The ElectronicProduct class extends the functionality of the

    Product class and includes additional attributes for brand and

    model.

    It inherits the methods from the Product class, including

    display\_information(), which is overridden to also display

    the brand and model.

    The ElectronicProduct class does not modify the input data

    and has no side effects.

    Methods:

        \_\_init\_\_(self, name, price, quantity, brand, model): Initializes a new

        instance of the ElectronicProduct class with the specified name,

        price, quantity, brand, and model.

        display\_information(self): Displays the name, price, quantity, brand,

        and model of the electronic product.

    '''

*def* \_\_init\_\_(*self*, *name*, *price*, *quantity*, *brand*, *model*):

        super().\_\_init\_\_(*name*, *price*, *quantity*)

*self*.brand = *brand*

*self*.model = *model*

*def* display\_information(*self*):

        super().display\_information()

        print("Brand is : ",*self*.brand)

        print("Model is : ",*self*.model)

*class* ClothingProduct(product):

    '''

    The ClothingProduct class represents a clothing product,

    which is a specialized type of Product. It inherits the attributes

    and methods from the Product class and adds additional attributes

    for size and color.

    The ClothingProduct class extends the functionality of the

    Product class and includes additional attributes for size and

    color.

    It inherits the methods from the Product class, including

    display\_information(), which is overridden to also display

    the size and color.

    The ClothingProduct class does not modify the input data

    and has no side effects.

    Methods:

        \_\_init\_\_(self, name, price, quantity, size, color): Initializes a new

        instance of the ClothingProduct class with the specified name,

        price, quantity, size, and color.

        display\_information(self):

            Displays the name, price, quantity, size, and color of the clothing

            product.

    '''

*def* \_\_init\_\_(*self*, *name*, *price*, *quantity*, *size*, *color*):

        super().\_\_init\_\_(*name*, *price*, *quantity*)

*self*.size = *size*

*self*.color = *color*

*def* display\_information(*self*):

        super().display\_information()

        print("Size is : ",*self*.size)

        print("Color is : ",*self*.color)

*def* add\_items\_clothing(*data\_dict*,*size*,*color*):

    '''

    The add\_items\_clothing function takes a dictionary containing data

    of clothing products, as well as lists for sizes and colors. It

    creates ClothingProduct objects based on the data and returns a

    list of these objects.

    The input is not modified in any way and there are no side effects.

    Parameters:

        data\_dict: A dictionary containing the data of clothing products.

        The keys are the names of the products, and the values are lists of

        [price, quantity].

        size: A list of sizes corresponding to each product in the data\_dict.

        color: A list of colors corresponding to each product in the data\_dict.

    Returns:

        A list of ClothingProduct objects created based on the data provided.

    '''

    keys = [x for x in *data\_dict*.keys()]

    clothing\_objects = []

    for key in range(len(keys)):

        name = keys[key]

        price, quantity = *data\_dict*[name]

        clothing\_objects.append(ClothingProduct(name,price,quantity,*size*[key],*color*[key]))

    return clothing\_objects

*def* add\_items\_electronic(*data\_dict*,*brand*,*model*):

    '''

    The add\_items\_electronic function takes a dictionary containing data of

    electronic products, as well as lists for brands and models. It creates

    ElectronicProduct objects based on the data and returns a list of these

    objects.

    Parameters:

        data\_dict: A dictionary containing the data of electronic products.

        The keys are the names of the products, and the values are lists of

        [price, quantity].

        brand: A list of brands corresponding to each product in the data\_dict.

        model: A list of models corresponding to each product in the data\_dict.

    Returns:

        A list of ElectronicProduct objects created based on the data provided.

    '''

    keys = [x for x in *data\_dict*.keys()]

    Electronic\_objects = []

    for key in range(len(keys)):

        name = keys[key]

        price, quantity = *data\_dict*[name]

        Electronic\_objects.append(ElectronicProduct(name,price,quantity,*brand*[key],*model*[key]))

    return Electronic\_objects

*def* input\_create(*data\_dict*):

    '''

    The input\_create function prompts the user to enter information

    for a new product and adds it to the provided data\_dict.

    Parameters:

        data\_dict: A dictionary containing the existing data of

        products.

    Returns:

        The updated data\_dict with the newly added product.

    '''

    name = input("Enter name : ")

    price = int(input("Enter price : "))

    quantity = int(input("Enter quantity : "))

    vals = [price,quantity]

*data\_dict*[name] = vals

    return *data\_dict*

#driver code

if \_\_name\_\_ == '\_\_main\_\_':

    #this part of the code will only be run when the function is called directly

    #it will not be executed when it is imported as a module

    a = ElectronicProduct("Phone",500,5,"Orange","EX1550")

    b = ClothingProduct("T-shirt",10000,1,"Medium","Blue")

    c = ClothingProduct("Shorts",15000,2,"Large","Black")

    a.display\_information()

    print()

    b.display\_information()

    print()

    b+c  #for printing combo offer

    print()

    data\_dict = {}

    brand\_names = []

    model\_names = []

    n = int(input("Enter number of entries electronic devices: "))

    for i in range(n):

        data\_dict = input\_create(data\_dict)

        brand = input("Enter brand name : ")

        model = input("Enter model name : ")

        brand\_names.append(brand)

        model\_names.append(model)

        print()

    electronic\_objects = add\_items\_electronic(data\_dict,brand\_names,model\_names)

    data\_dict = {}

    sizes = []

    colors = []

    n = int(input("Enter number of entries for clothes: "))

    for i in range(n):

        data\_dict = input\_create(data\_dict)

        size = input("Enter size : ")

        color = input("Enter color name : ")

        sizes.append(size)

        colors.append(color)

        print()

    clothing\_objects = add\_items\_clothing(data\_dict,sizes,colors)

    for electronic\_object in electronic\_objects:

        electronic\_object.display\_information()

        print()

    for Clothing\_object in clothing\_objects:

        Clothing\_object.display\_information()

        print()

III. OUTPUT:

Name is : Phone

Price is : 500

Quantity is : 5

Brand is : Orange

Model is : EX1550

Name is : T-shirt

Price is : 10000

Quantity is : 1

Size is : Medium

Color is : Blue

Combo offer for T-shirt and Shorts !

For a price of 40000

Enter number of entries electronic devices: 1

Enter name : Tablet

Enter price : 10000

Enter quantity : 1

Enter model name : ex10992

Enter number of entries for clothes: 1

Enter name : Jeans

Enter price : 1000

Enter quantity : 2

Enter size : Large

Enter color name : Blue

Name is : Tablet

Price is : 10000

Quantity is : 1

Brand is : Orange

Model is : ex10992

Name is : Jeans

Price is : 1000

Quantity is : 2

Size is : Large

Color is : Blue