# Port Folio Milestone

**Online Shopping Cart**

This application builds upon the application started in Module 4. The ItemsToPurchase class created in Module 4 is used for containing an item to purchase. In Module 6, a new class is created, the ShoppingCart class. This class contains the following attributes:

* customer\_name: string
* current\_date: string

The ShoppingCart class accepts the following parameter when created:

* ItemToPurchase: ItemsToPurchase

The ShoppingCart class implements the following methods:

* add\_item:
  + Adds an item to cart\_items list. Has parameter ItemToPurchase. Does not return anything.
* remove\_item():
  + Removes item from cart\_items list. Has a string (an item's name) parameter. Does not return anything.
  + If item name cannot be found, output this message: Item not found in cart. Nothing removed.
* modify\_item()
  + Modifies an item's description, price, and/or quantity. Has parameter ItemToPurchase. Does not return anything.
  + If item can be found (by name) in cart, check if parameter has default values for description, price, and quantity. If not, modify item in cart.
  + If item cannot be found (by name) in cart, output this message: Item not found in cart. Nothing modified.
* get\_num\_items\_in\_cart()
  + Returns quantity of all items in cart. Has no parameters.
* get\_cost\_of\_cart()
  + Determines and returns the total cost of items in cart. Has no parameters.
* print\_total()
  + Outputs total of objects in cart.
  + If cart is empty, output this message: SHOPPING CART IS EMPTY
* print\_descriptions()
  + Outputs each item's description.

In the main section of the code, a function named print\_menu is defined. This function accepts the following parameter:

* ShoppingCart: ShoppingCart

The function print\_menu outputs a menu of options to manipulate the shopping cart. Each option is represented by a single character.

MENU  
a - Add item to cart  
r - Remove item from cart  
c - Change item quantity  
i - Output items' descriptions  
o - Output shopping cart  
q - Quit  
Choose an option:

The function continues until the user enters a q to Quit. For Module 6, only the “i - Output items' descriptions” and “o - Output shopping cart” menu choices are implemented.

* “i - Output items' descriptions”
  + Executes ShoppingCart.print\_descriptions()
* “o - Output shopping cart”
  + Executes ShoppingCart.print\_total()

The submitted Python code demonstrates functionality by creating a ShoppingCart object and then prompting the user for the customer name and current date. Next, an ItemsToPurchase object is created. A while loop allows the user to enter the item name, item description, item price, and item quantity of an items and adds the object to the ShoppingCart object. Additional ItemsToPurchase can be added to the ShoppingCart boject until the user enters a single c character. Upon the completion of adding these initial items, the print\_menu function is called. The user can then choose from the menu or enter a single q character to quit.

## Python Code

# Define ItemsToPurchase class

class ItemsToPurchase:

item\_name = ''

item\_price = 0.0

item\_quantity = 0

def \_\_init\_\_(self,item\_name = 'none',item\_description = 'none',item\_price = 0,item\_quantity = 0):

self.item\_name = item\_name

self.item\_description = item\_description

self.item\_price = item\_price

self.item\_quantity = item\_quantity

def print\_item\_cost(self):

print('{} {} @${:.2f} = ${:.2f}'.format(self.item\_name,self.item\_quantity,self.item\_price,self.item\_price\*self.item\_quantity))

# Define ShoppingCart class

class ShoppingCart: #Parameterized constructor, which takes the customer name and date as parameters

customer\_name = 'none'

current\_date = 'January 1, 2020'

def \_\_init\_\_(self,customer\_name = 'none',current\_date = 'January 1, 2020'):

self.customer\_name = customer\_name #customer\_name (string) - Initialized in default constructor to "none"

self.current\_date = current\_date #current\_date (string) - Initialized in default constructor to "January 1, 2020"

self.cart\_items = [] #cart\_items (list)

def add\_item(self,ItemsToPurchase): #Adds an item to cart\_items list. Has parameter ItemToPurchase. Does not return anything.

self.cart\_items.append(ItemsToPurchase)

def remove\_item(self,item\_name): #Removes item from cart\_items list. Has a string (an item's name) parameter. Does not return anything.

index = 0

found = False

for cart\_item in self.cart\_items:

if item\_name == cart\_item.item\_name:

del self.cart\_items[index]

found = True

break

index += 1

if not found:

print('Item not found in cart. Nothing removed.') #If item name cannot be found, output this message: Item not found in cart. Nothing removed.

def modify\_item(self,ItemToPurchase): #Modifies an item's description, price, and/or quantity. Has parameter ItemToPurchase. Does not return anything.

index = 0

found = False

temp = ItemsToPurchase()

for cart\_item in self.cart\_items:

if ItemToPurchase.item\_name == cart\_item.item\_name: #If item can be found (by name) in cart,

if ItemToPurchase.item\_description != temp.item\_description: #check if parameter has default values for description, price, and quantity.

self.cart\_items[index].item\_description = ItemToPurchase.item\_description #If not, modify item in cart

if ItemToPurchase.item\_price != temp.item\_price: #check if parameter has default values for description, price, and quantity.

self.cart\_items[index].item\_price = ItemToPurchase.item\_price #If not, modify item in cart

if ItemToPurchase.item\_quantity != temp.item\_quantity: #check if parameter has default values for description, price, and quantity.

self.cart\_items[index].item\_quantity = ItemToPurchase.item\_quantity #If not, modify item in cart

found = True

break

index += 1

if not found: #If item cannot be found (by name) in cart

print('Item not found in cart. Nothing modified.') #output this message: Item not found in cart. Nothing modified.

def get\_num\_items\_in\_cart(self): #Returns quantity of all items in cart. Has no parameters

num\_items = 0

for cart\_item in self.cart\_items:

num\_items = num\_items + cart\_item.item\_quantity

return num\_items

def get\_cost\_of\_cart(self): #Determines and returns the total cost of items in cart. Has no parameters

total\_cost = 0

for cart\_item in self.cart\_items:

total\_cost = total\_cost + cart\_item.item\_price \* cart\_item.item\_quantity

return total\_cost

def print\_total(self): #Outputs total of objects in cart.

print('OUTPUT SHOPPING CART')

if self.get\_num\_items\_in\_cart() > 0:

print('{}\'s Shopping Cart - {}'.format(self.customer\_name,self.current\_date))

print('Number of Items: {}'.format(self.get\_num\_items\_in\_cart()))

for cart\_item in self.cart\_items:

cart\_item.print\_item\_cost()

print('Total: ${:.2f}'.format(self.get\_cost\_of\_cart()))

else:

print('SHOPPING CART IS EMPTY') #If cart is empty, output this message: SHOPPING CART IS EMPTY

def print\_descriptions(self): #Outputs each item's description

print('OUTPUT ITEMS\'S DESCRIPTIONS')

print('{}\'s Shopping Cart - {}'.format(self.customer\_name,self.current\_date))

for cart\_item in self.cart\_items:

print('{}: {}'.format(cart\_item.item\_name,cart\_item.item\_description))

#Step 5: In the main section of your code, implement the print\_menu() function. print\_menu() has a ShoppingCart

# arameter and outputs a menu of options to manipulate the shopping cart. Each option is represented by a single

# character. Build and output the menu within the function

def print\_menu(ShoppingCart):

menu\_choice = ''

while menu\_choice != 'q':

print('MENU')

print('a - Add item to cart')

print('r - Remove item from cart')

print('c - Change item quantity')

print('i - Output items\' descriptions')

print('o - Output shopping cart')

print('q - Quit')

menu\_choice = input('Choose an option: ')

if menu\_choice == 'a':

print(menu\_choice)

elif menu\_choice == 'r':

print(menu\_choice)

elif menu\_choice == 'c':

print(menu\_choice)

elif menu\_choice == 'i':

ShoppingCart.print\_descriptions() #Step 6: Implement Output shopping cart menu option. Implement Output item's description menu option

elif menu\_choice == 'o':

ShoppingCart.print\_total() # Steo 6: Implement Output shopping cart menu option

if \_\_name\_\_ == "\_\_main\_\_":

#Create ShoppingCart

ShoppingCart = ShoppingCart()

#Prompt user for customer information

ShoppingCart.customer\_name = input('Input customer name:')

ShoppingCart.current\_date = input('Input current date:')

#Create ItemsToPurchase

Item = ItemsToPurchase()

num\_item = 1

# Gather shopping cart entries until the user enters a 'q'

# Initialize Sentinel Character

user\_input = ''

while user\_input != 'c':

print('Item {}'.format(num\_item))

Item.item\_name = input('Enter the item name:')

Item.item\_description = input('Enter item description:')

Item.item\_price = float(input('Enter the item price:'))

Item.item\_quantity = int(input('Enter the item quantity:'))

ShoppingCart.add\_item(Item)

user\_input = input('Enter c to continue, any other key to add another item: ')

num\_item += 1

print\_menu(ShoppingCart)

## Output

A screenshot of a computer program

Description automatically generated

# Git-repo

<https://github.com/tlerunner/git-repo/tree/main/Module%206>