**Portfolio Project: Online Shopping Cart**

From Milestone 1:

Step 1:

Build the ItemToPurchase class with the following specifications:

Attributes

item\_name (string)

item\_price (float)

item\_quantity (int)

Default constructor

Initializes item's name = "none", item's price = 0, item's quantity = 0

Method

print\_item\_cost()

Example of print\_item\_cost() output:

Bottled Water 10 @ $1 = $10

Step 2:

In the main section of your code, prompt the user for two items and create two objects of the ItemToPurchase class.

Example:

Item 1

Enter the item name:

Chocolate Chips

Enter the item price:

3

Enter the item quantity:

1

Item 2

Enter the item name:

Bottled Water

Enter the item price:

1

Enter the item quantity:

10

Step 3:

Add the costs of the two items together and output the total cost.

Example:

TOTAL COST

Chocolate Chips 1 @ $3 = $3

Bottled Water 10 @ $1 = $10

Total: $13

Fix any issues from milestone 1 submission prior to submitting the Portfolio Project.

From Milestone 2:

Step 4:

Build the ShoppingCart class with the following data attributes and related methods. Note: Some can be method stubs (empty methods) initially, to be completed in later steps.

Parameterized constructor, which takes the customer name and date as parameters

Attributes

customer\_name (string) - Initialized in default constructor to "none"

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

cart\_items (list)

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.

Example of print\_total() output:

John Doe's Shopping Cart - February 1, 2020

Number of Items: 8

Nike Romaleos 2 @ $189 = $378

Chocolate Chips 5 @ $3 = $15

Powerbeats 2 Headphones 1 @ $128 = $128

Total: $521

Example of print\_descriptions() output:

John Doe's Shopping Cart - February 1, 2020

Item Descriptions

Nike Romaleos: Volt color, Weightlifting shoes

Chocolate Chips: Semi-sweet

Powerbeats 2 Headphones: Bluetooth headphones

Step 5:

In the main section of your code, implement the print\_menu() function. print\_menu() has a ShoppingCart parameter 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.

If an invalid character is entered, continue to prompt for a valid choice. Hint: Implement Quit before implementing other options. Call print\_menu() in the main() function. Continue to execute the menu until the user enters q to Quit.

Example:

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:

Step 6:

Implement Output shopping cart menu option. Implement Output item's description menu option.

Example of shopping cart menu option:

OUTPUT SHOPPING CART

John Doe's Shopping Cart - February 1, 2020

Number of Items: 8

Nike Romaleos 2 @ $189 = $378

Chocolate Chips 5 @ $3 = $15

Powerbeats 2 Headphones 1 @ $128 = $128

Total: $521

Example of item description menu option.

OUTPUT ITEMS' DESCRIPTIONS

John Doe's Shopping Cart - February 1, 2020

Item Descriptions

Nike Romaleos: Volt color, Weightlifting shoes

Chocolate Chips: Semi-sweet

Powerbeats 2 Headphones: Bluetooth headphones

Fix any issues from milestone 2 submission prior to submitting the Portfolio Project.

Additional tasks for the final project submission:

Step 7:

In the main section of your code, prompt the user for a customer's name and today's date. Output the name and date. Create an object of type ShoppingCart.

Example:

Enter customer's name:

John Doe

Enter today's date:

February 1, 2020

Customer name: John Doe

Today's date: February 1, 2020

Step 8:

Implement Add item to cart menu option.

Example:

ADD ITEM TO CART

Enter the item name:

Nike Romaleos

Enter the item description:

Volt color, Weightlifting shoes

Enter the item price:

189

Enter the item quantity:

2

Step 9:

Implement remove item menu option.

Example:

REMOVE ITEM FROM CART

Enter name of item to remove:

Chocolate Chips

Step 10:

Implement Change item quantity menu option. Hint: Make new ItemToPurchase object before using ModifyItem() method.

Example:

CHANGE ITEM QUANTITY

Enter the item name:

Nike Romaleos

Enter the new quantity:

3

Compile and submit your pseudocode, source code, screenshots of the application executing the code, the results and GIT repository in a single document (Word is preferred).

**Pseudocode components Explanation:**

1. **main**:
   * Prompts the user for the customer's name and the date.
   * Creates a ShoppingCart object.
   * Calls the print\_menu function to start the shopping interaction.
2. **print\_menu**:
   * Continuously displays the menu with options (add, remove, change, output descriptions, output cart, quit).
   * Depending on the user's choice, it handles the appropriate functionality (adding/removing items, changing quantities, etc.).
3. **add\_item**:
   * Adds an item to the shopping cart.
4. **remove\_item**:
   * Removes an item from the cart by its name.
5. **modify\_item**:
   * Modifies the quantity (and other details if necessary) of an item in the cart.
6. **print\_total**:
   * Outputs the total cost and the itemized list of the shopping cart.
7. **print\_descriptions**:
   * Displays descriptions of all items in the cart.
8. **Loop**:
   * The print\_menu function runs in a loop, continuously asking for user input until the user decides to quit by choosing 'q'.

**Compiled PSEUDOCODE:**

START

FUNCTION main:

# Step 7: Get customer details

Prompt user to enter customer's name

Prompt user to enter today's date

Output customer's name and today's date

# Create a shopping cart object with customer details

Create ShoppingCart object with customer's name and today's date

# Call the print\_menu function to start the shopping session

CALL print\_menu(cart)

END FUNCTION

FUNCTION print\_menu(cart):

WHILE True:

PRINT "MENU"

PRINT available options:

a - Add item to cart

r - Remove item from cart

c - Change item quantity

i - Output items' descriptions

o - Output shopping cart

q - Quit

Prompt user for an option

IF user chooses 'a' (Add item to cart):

PRINT "ADD ITEM TO CART"

Prompt user for item name, description, price, and quantity

Create a new ItemToPurchase object with the entered details

CALL add\_item on cart object with the new item

PRINT confirmation that item was added to the cart

ELSE IF user chooses 'r' (Remove item from cart):

PRINT "REMOVE ITEM FROM CART"

Prompt user for the name of the item to remove

CALL remove\_item on cart object with the item name

ELSE IF user chooses 'c' (Change item quantity):

PRINT "CHANGE ITEM QUANTITY"

Prompt user for item name and the new quantity

Create a new ItemToPurchase object with the new quantity

CALL modify\_item on cart object with the updated item details

ELSE IF user chooses 'i' (Output items' descriptions):

PRINT "OUTPUT ITEMS' DESCRIPTIONS"

CALL print\_descriptions on cart object

ELSE IF user chooses 'o' (Output shopping cart):

PRINT "OUTPUT SHOPPING CART"

CALL print\_total on cart object

ELSE IF user chooses 'q' (Quit):

PRINT "Exiting the shopping cart program."

BREAK loop (Exit the menu)

ELSE:

PRINT "Invalid choice. Please choose a valid option."

END WHILE

END FUNCTION

FUNCTION add\_item(cart, item):

Append the item to the cart's item list

END FUNCTION

FUNCTION remove\_item(cart, item\_name):

Search for item by name in the cart's item list

IF item is found:

Remove the item from the cart

PRINT that the item has been removed

ELSE:

PRINT that the item was not found

END FUNCTION

FUNCTION modify\_item(cart, item):

Search for item by name in the cart's item list

IF item is found:

Update the item's details (description, price, quantity)

PRINT that the item has been updated

ELSE:

PRINT that the item was not found

END FUNCTION

FUNCTION print\_total(cart):

PRINT customer’s name and current date

PRINT the total number of items in the cart

IF cart is empty:

PRINT "SHOPPING CART IS EMPTY"

ELSE:

For each item in cart:

CALL print\_item\_cost to display item details (name, quantity, price, total cost)

PRINT the total cost of the entire cart

END FUNCTION

FUNCTION print\_descriptions(cart):

PRINT customer’s name and current date

PRINT "Item Descriptions"

IF cart is empty:

PRINT "No items in the cart."

ELSE:

For each item in cart:

PRINT the item’s name and description

END FUNCTION

END

Here is the compiled final code combining all parts. This includes the ItemToPurchase class, the ShoppingCart class, and the main function which prompts for input and uses the print\_menu function to manage the shopping cart.

**Compiled final code**

# ItemToPurchase class

class ItemToPurchase:

def \_\_init\_\_(self, item\_name="none", item\_price=0.0, item\_quantity=0, item\_description="none"):

self.item\_name = item\_name

self.item\_price = item\_price

self.item\_quantity = item\_quantity

self.item\_description = item\_description

def print\_item\_cost(self):

total\_cost = self.item\_price \* self.item\_quantity

print(f"{self.item\_name} {self.item\_quantity} @ ${self.item\_price} = ${total\_cost}")

# ShoppingCart class

class ShoppingCart:

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

self.customer\_name = customer\_name

self.current\_date = current\_date

self.cart\_items = []

def add\_item(self, item):

self.cart\_items.append(item)

def remove\_item(self, item\_name):

for item in self.cart\_items:

if item.item\_name == item\_name:

self.cart\_items.remove(item)

print(f"{item\_name} has been removed from the cart.\n")

return

print("Item not found in cart. Nothing removed.\n")

def modify\_item(self, item):

for cart\_item in self.cart\_items:

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

if item.item\_description != "none":

cart\_item.item\_description = item.item\_description

if item.item\_price != 0.0:

cart\_item.item\_price = item.item\_price

if item.item\_quantity != 0:

cart\_item.item\_quantity = item.item\_quantity

print(f"{item.item\_name} has been updated.\n")

return

print("Item not found in cart. Nothing modified.\n")

def get\_num\_items\_in\_cart(self):

return sum(item.item\_quantity for item in self.cart\_items)

def get\_cost\_of\_cart(self):

return sum(item.item\_price \* item.item\_quantity for item in self.cart\_items)

def print\_total(self):

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

num\_items = self.get\_num\_items\_in\_cart()

print(f"Number of Items: {num\_items}")

if num\_items == 0:

print("SHOPPING CART IS EMPTY\n")

else:

total\_cost = 0

for item in self.cart\_items:

item.print\_item\_cost()

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

print(f"Total: ${total\_cost}\n")

def print\_descriptions(self):

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

print("Item Descriptions")

if not self.cart\_items:

print("No items in the cart.\n")

for item in self.cart\_items:

print(f"{item.item\_name}: {item.item\_description}")

# Main function to interact with the user

def main():

# Step 7: Prompt the user for customer's name and today's date

customer\_name = input("Enter customer's name: ")

current\_date = input("Enter today's date: ")

# Output the customer's name and today's date

print(f"Customer name: {customer\_name}")

print(f"Today's date: {current\_date}")

# Create a ShoppingCart object with the provided customer name and date

cart = ShoppingCart(customer\_name, current\_date)

# Call the print\_menu function to start the shopping session

print\_menu(cart)

# Menu function to display options and handle user choices

def print\_menu(cart: ShoppingCart):

while True:

print("\nMENU")

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")

# Get the user's choice

choice = input("Choose an option: ")

if choice == 'a':

# Add item to cart

print("\nADD ITEM TO CART")

item\_name = input("Enter the item name: ")

item\_description = input("Enter the item description: ")

item\_price = float(input("Enter the item price: "))

item\_quantity = int(input("Enter the item quantity: "))

# Create an ItemToPurchase object with the entered data

item = ItemToPurchase(item\_name, item\_price, item\_quantity)

item.item\_description = item\_description # Assign the description

# Add the item to the shopping cart

cart.add\_item(item)

print(f"{item\_name} has been added to the cart.\n")

elif choice == 'r':

# Remove item from cart

print("\nREMOVE ITEM FROM CART")

item\_name = input("Enter the name of the item to remove: ")

cart.remove\_item(item\_name)

elif choice == 'c':

# Change item quantity

print("\nCHANGE ITEM QUANTITY")

item\_name = input("Enter the item name: ")

new\_quantity = int(input("Enter the new quantity: "))

# Modify the item quantity in the cart

item = ItemToPurchase(item\_name, 0.0, new\_quantity) # Only change the quantity

cart.modify\_item(item)

elif choice == 'i':

# Output items' descriptions

print("\nOUTPUT ITEMS' DESCRIPTIONS")

cart.print\_descriptions()

elif choice == 'o':

# Output shopping cart

print("\nOUTPUT SHOPPING CART")

cart.print\_total()

elif choice == 'q':

# Quit the program

print("Exiting the shopping cart program.")

break

else:

# Handle invalid input

print("Invalid choice. Please choose a valid option.")

# Run the main function

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

main()

**Explanation of Code:**

1. **ItemToPurchase Class:**
   * It represents an item with attributes like name, price, quantity, and description.
   * The print\_item\_cost() method calculates and displays the total cost of an item in the cart based on its quantity and price.
2. **ShoppingCart Class:**
   * This class holds a list of ItemToPurchase objects and provides several methods to add, remove, and modify items.
   * add\_item() adds an item to the cart.
   * remove\_item() removes an item from the cart by its name.
   * modify\_item() modifies the details of an existing item (like quantity).
   * print\_total() outputs the cart's total cost.
   * print\_descriptions() outputs the descriptions of all items in the cart.
3. **Main Function:**
   * Prompts the user for the customer’s name and date.
   * Creates a ShoppingCart object with the provided information.
   * Calls the print\_menu() function to start the shopping session.
4. **Menu Options:**
   * The user can choose to add an item, remove an item, change an item’s quantity, print descriptions, print the entire cart, or quit the program.
   * The print\_menu() function repeatedly displays the menu until the user opts to quit.

**Screenshots of compiled code and execution**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated