**CSC500**

**Module 8 Portfolio Project**

**GIT Repository:** <https://github.com/focomapper/CSC500>

**Introduction:**

Module 8 was the final addition of functionality to the portfolio project. Additional code required implementing the Add item, Remove item and Change item quantity methods in the ShoppingCart class. There were some inconsistencies between the Module 6 milestone and Module 8 milestone instructions regarding the Change item method. Module 6 required the method to change the item’s description, price, and/or quantity where Module 8 only instructed to change the item’s quantity. I decided that since Module 8 was the most recent set of instructions, it was the most accurate and I designed the Change item method to only modify an item’s quantity.

Furthermore, Module 6 instructed to check if the input item’s parameters had default values for description, price, and quantity. And if not, modify the item in the cart. Since only the item quantity was being modified and due to the nature of how the items were being added to the cart where the user was entering the name, description, price, and quantity for each item, the default values would never occur unless the user intentionally entered them, which didn’t make sense. As a result, I disregarded that check.

I found this exercise more challenging than the other module critical thinking assignments and thinking about every use case caused me to make the code a little more complex. For example, since item names were being used to search for items in the cart, there needed to be a check to be sure item names were unique. I added some logic to the Add item method to prevent the user from adding a duplicate name, otherwise the remove item and change item methods would not work properly as they would remove or change the first item encountered in the shopping cart with the qualifying name. This problem could have been remedied by identifying items in the cart using both name and description but then the Remove item method would have to be modified to use ItemToPuchase as a parameter instead of just the item name. The Add item method would also have to be modified to ensure the item’s name and description together were unique. I decided to keep it simple and just ensure the item name was unique.

**Pseudo code for entire project-**

1. Import datetime library.
2. Create ItemToPurchase class.
   1. Implement default constructor and print item cost method.
3. Create ShoppingCart class.
   1. Implement constructor, add\_item, get\_num\_items\_in\_cart, get\_cost\_of\_cart, print\_total and print\_descriptions methods.
   2. Implement print\_menu, add, remove, change, output shopping cart and quit methods.
4. Prompt user to add name and date.
5. Run ShoppingCart class print\_menu method and respond to user input.
6. Print results.

**Source code for entire project-**

from datetime import datetime  
  
  
# Create Item To Purchase class  
class ItemToPurchase:  
 def \_\_init\_\_(self):  
 # Initialize variables  
 self.item\_name = 'none'  
 self.item\_price = float(0)  
 self.item\_quantity = 0  
 self.item\_description = 'none' # Assumed this is necessary but not in instructions  
  
 # Print item method  
 def print\_item\_cost(self):  
 total\_price = self.item\_price \* self.item\_quantity  
 print('{:^50}'.format(self.item\_name + ' ' + str(self.item\_quantity) + ' @ ${:,.2f}'.format(self.item\_price) + ' = ${:,.2f}'.format(total\_price)))  
  
  
# Create Shopping Cart class  
class ShoppingCart:  
  
 cart\_contents = []  
  
 def \_\_init\_\_(self, name='none', date="January 1, 2020"):  
 self.customer\_name = name  
 self.current\_date = date  
  
 def add\_item(self, item): # item to purchase  
 # check if name is unique, if so prompt user to enter unique name  
 item\_exists = False  
 for existing\_item in self.cart\_contents:  
 if existing\_item.item\_name == item.item\_name:  
 item\_exists = True  
 if item\_exists:  
 print('{:^50}'.format('Item with name ' + item.item\_name + ' already exists in cart. Please use a different name.'))  
 else:  
 self.cart\_contents.append(item)  
 print('{:^50}'.format('Item ' + item.item\_name + ' added to shopping cart'))  
  
 def remove\_item(self, item\_name): # item name  
 item\_found = False  
 for item in self.cart\_contents:  
 if item.item\_name == item\_name:  
 self.cart\_contents.remove(item)  
 item\_found = True  
 print('{:^50}'.format('Item ' + item.item\_name + ' removed from shopping cart'))  
 if not item\_found:  
 print('{:^50}'.format('Item not found in cart. Nothing removed'))  
  
 def modify\_item(self, item\_to\_modify): # item to purchase  
 item\_in\_cart = False  
 for item in self.cart\_contents:   
 if item\_to\_modify.item\_name == item.item\_name:  
 print('{:^50}'.format('Item ' + item.item\_name + ' quantity changed from ' + str(item.item\_quantity) + ' to ' + str(item\_to\_modify.item\_quantity) + '.'))  
 item.item\_quantity = item\_to\_modify.item\_quantity  
 item\_in\_cart = True  
 if not item\_in\_cart:  
 print('{:^50}'.format('Item not found in cart. Nothing changed'))  
  
 def get\_num\_items\_in\_cart(self):  
 total\_quantity = 0  
 for item in self.cart\_contents:  
 total\_quantity += item.item\_quantity  
 return total\_quantity  
   
 def get\_cost\_of\_cart(self):  
 cost = 0  
 for item in self.cart\_contents:  
 cost += item.item\_price \* item.item\_quantity  
 return cost  
   
 def print\_total(self):  
 print('{:^50}'.format(self.customer\_name + '\'s Shopping Cart - ' + self.current\_date))  
 if len(self.cart\_contents) == 0:  
 print('{:^50}'.format('SHOPPING CART IS EMPTY'))  
 else:  
 for item in self.cart\_contents:  
 item.print\_item\_cost()  
 print('{:^50}'.format('Number of Items: ' + str(self.get\_num\_items\_in\_cart())))  
 print('{:^50}'.format('${:,.2f}'.format(self.get\_cost\_of\_cart())))  
  
 def print\_description(self):  
 print('{:^50}'.format(self.customer\_name + '\'s Shopping Cart - ' + self.current\_date))  
 print('{:^50}'.format('Item Descriptions'))  
 if self.get\_num\_items\_in\_cart() == 0:  
 print('{:^50}'.format('No items in cart'))  
 else:  
 for item in self.cart\_contents:  
 print('{:^50}'.format(item.item\_name + ': ' + item.item\_description))  
  
  
def print\_menu(cart):  
 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')  
 option = str(input('Choose an option: '))  
 if option == 'q':  
 return option  
 elif option == 'a':  
 print('{:^50}'.format('ADD ITEM TO CART'))  
 item = ItemToPurchase()  
 item.item\_name = str(input('Enter the item name: '))  
 item.item\_price = float(input('Enter the item price: '))  
 while not type(item.item\_price) is float:  
 item.item\_price = float(input('Enter the item price in dollar and cents: '))  
 item.item\_quantity = int(input('Enter the item quantity: '))  
 while not type(item.item\_quantity) is int:  
 item.item\_quantity = int(input('Enter the item quantity in whole numbers: '))  
 item.item\_description = str(input('Enter the item description: '))  
 if item.item\_quantity > 0:  
 cart.add\_item(item)  
 else:  
 print('{:^50}'.format('Item quantity must be greater than 0.'))  
 print('{:^50}'.format('Item not added to shopping cart.'))  
 elif option == 'r':  
 print('{:^50}'.format('REMOVE ITEM FROM CART'))  
 item\_name\_to\_remove = str(input('Enter the item name to remove: '))  
 cart.remove\_item(item\_name\_to\_remove)  
 elif option == 'c':  
 print('{:^50}'.format('CHANGE ITEM QUANTITY'))  
 item = ItemToPurchase()  
 item.item\_name = str(input('Enter the item name to modify: '))  
 try:  
 item.item\_quantity = int(input('Enter new quantity: '))  
 except ValueError:  
 print('{:^50}'.format('Quantity number must be a whole number. Please try again.'))  
 if item.item\_quantity > 0:  
 cart.modify\_item(item)  
 else:  
 print('{:^50}'.format('Item quantity must be great than 0.'))  
 print('{:^50}'.format('Please try again or use remove option if item no longer needed.'))  
 del item  
 elif option == 'i':  
 cart.print\_description()  
 elif option == 'o':  
 cart.print\_total()  
  
  
customer\_name = str(input('Enter customer name: ')).title()  
while True:  
 cart\_date = input('Enter today\'s date: month day, year (ex. January 1, 2024): ')  
 try:  
 valid\_date = datetime.strptime(cart\_date, '%B %d, %Y')  
 formatted\_date = str(valid\_date.strftime('%B %d, %Y'))  
 break  
 except ValueError:  
 print('{:^50}'.format('Date is invalid, please enter valid date'))  
print(f'Customer name: {customer\_name}')  
print(f'today\'s date: {cart\_date}')  
  
customer\_cart = ShoppingCart(customer\_name, formatted\_date)  
edit\_cart = ""  
while edit\_cart != 'q':  
 edit\_cart = print\_menu(customer\_cart)  
  
customer\_cart.print\_total()  
customer\_cart.print\_description()

**Screenshots of code executing-**

**A screenshot of a computer program

Description automatically generated**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated