Romeo Sayon

Final project

Gibhub:

#romeo Sayon

#CSU Global

# Portfolio Milestone/Online Shopping Cart

# November 04, 2023

class ItemToPurchase:

def \_\_init\_\_(self, name="none", price=0.0, quantity=0, description="none"):

self.name = name

self.price = price

self.quantity = quantity

self.description = description

def print\_item\_cost(self):

print(f"{self.name} {self.quantity} @ ${self.price} = ${self.price \* self.quantity}")

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):

found = False

for item in self.cart\_items:

if item.name == item\_name:

self.cart\_items.remove(item)

found = True

break

if not found:

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

def modify\_item(self, item):

found = False

for cart\_item in self.cart\_items:

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

if item.price != 0.0:

cart\_item.price = item.price

if item.quantity != 0:

cart\_item.quantity = item.quantity

if item.description != "none":

cart\_item.description = item.description

found = True

break

if not found:

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

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

total\_quantity = sum(item.quantity for item in self.cart\_items)

return total\_quantity

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

total\_cost = sum(item.price \* item.quantity for item in self.cart\_items)

return total\_cost

def print\_total(self):

if not self.cart\_items:

print("SHOPPING CART IS EMPTY")

else:

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

print(f"Number of Items: {self.get\_num\_items\_in\_cart()}\n")

for item in self.cart\_items:

item.print\_item\_cost()

print(f"\nTotal: ${self.get\_cost\_of\_cart()}")

def print\_descriptions(self):

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

for item in self.cart\_items:

print(f"{item.name}: {item.description}")

def print\_menu(cart):

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")

def output\_shopping\_cart(cart):

print("OUTPUT SHOPPING CART")

cart.print\_total()

def output\_item\_descriptions(cart):

print("OUTPUT ITEMS' DESCRIPTIONS")

cart.print\_descriptions()

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

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

purchase\_date = input("Enter today's date (e.g., YYYY-MM-DD): ")

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

print(f"Customer's Name: {cart.customer\_name}")

print(f"Today's Date: {cart.current\_date}")

while True:

print\_menu(cart)

choice = input("Choose an option: ")

if choice == "a":

item = ItemToPurchase(input("Enter the item name: "),

float(input("Enter the item price: ")),

int(input("Enter the item quantity: ")),

input("Enter the item description: "))

cart.add\_item(item)

elif choice == "r":

item\_name = input("REMOVE ITEM FROM CART\nEnter name of item to remove: ")

cart.remove\_item(item\_name)

elif choice == "c":

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

new\_item = ItemToPurchase(input("Enter the new item name: "),

float(input("Enter the new item price (or 0 to keep the current price): ")),

int(input("Enter the new item quantity (or 0 to keep the current quantity): ")),

input("Enter the new item description (or 'none' to keep the current description): "))

cart.modify\_item(new\_item)

elif choice == "i":

output\_item\_descriptions(cart)

elif choice == "o":

output\_shopping\_cart(cart)

elif choice == "q":

break

else:

print("Invalid option. Please try again.")

**step 8:**

**A screenshot of a computer

Description automatically generated**

**Step 9**

**A screenshot of a computer

Description automatically generated**

**Step 10**

**A close-up of a computer screen

Description automatically generated**