**Module 6 Portfolio Milestone Project**

Shashi B. Thakur

Colorado State University Global

CSC500: Principles of Programming

Dr. Steven Evans

March 02, 2024

**Module 6 Portfolio Milestone Project**

**Git Hub Link –**

<https://github.com/shashithakurcsu/CSUProjects/blob/main/Module5/rainfall_calculator.py>

<https://github.com/shashithakurcsu/CSUProjects/blob/main/Module5/bookclub_award.py>

This assignment is to enhance the Shopping Cart project we did for Module 4 assignment. The task is to develop set of classes that simulates the functionality of a shopping cart. The functionality should include adding, removing, and modifying items, as well as calculating the total cost and providing item descriptions.

Below is the detailed description of the solution including the overall solution comprises of structure of classes and logic behind the user interaction with a menu driven interface.

**ShoppingCart Class** – This class should have a parameterized constructor which takes the customer name and date as parameters.

**Attributes –**

**customer\_name** (string): The name of the customer owning the cart.

**current\_date** (string): The current date, representing when the shopping session took place.

**cart\_items** (list): A list of **ItemToPurchase** objects in the cart.

Code Snippet

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 = []

**Methods**:

* + - **\_\_init\_\_**: Initializes a new shopping cart with the customer's name, current date, and an empty list of items.
    - **add\_item**: Adds a new **ItemToPurchase** object to **cart\_items**.

def add\_item(self, item\_to\_purchase):

self.cart\_items.append(item\_to\_purchase)

* + - **remove\_item**: Removes an item from **cart\_items** by item name. If the item is not found, it prints a message to update user accordingly.
* def remove\_item(self, item\_name):
* found = False
* for item in self.cart\_items:
* if item.item\_name == item\_name:
* self.cart\_items.remove(item)
* found = True
* break
* if not found:
* print("Item not found in cart. Nothing removed.")
  + - **modify\_item**: Updates the details of an item in the cart if it exists. If the item does not exist, it prints a message to update user accordingly.
* def modify\_item(self, item\_to\_purchase):
* item\_found = False
* # Iterate through the items in the cart to find a match by name
* for item in self.cart\_items:
* if item.item\_name == item\_to\_purchase.item\_name:
* item\_found = True # Mark as found
* # Check for non-default values and modify the item accordingly
* # Assuming default for description is "none", price is 0.0, and quantity is 0
* if item\_to\_purchase.item\_description != "none":
* item.item\_description = item\_to\_purchase.item\_description
* if item\_to\_purchase.item\_price != 0.0:
* item.item\_price = item\_to\_purchase.item\_price
* if item\_to\_purchase.item\_quantity != 0:
* item.item\_quantity = item\_to\_purchase.item\_quantity
* break # Exit the loop as we've found and processed the item
* if not item\_found:
* print("Item not found in cart. Nothing modified.")
  + - **get\_num\_items\_in\_cart**: Returns the total quantity of all items in the cart.
* def get\_num\_items\_in\_cart(self):
* total\_quantity = sum(item.item\_quantity for item in self.cart\_items)
* return total\_quantity
  + - **get\_cost\_of\_cart**: Calculates and returns the total cost of the items in the cart.
* def get\_cost\_of\_cart(self):
* total\_cost = 0 # Initialize total cost
* # Iterate over each item in the cart
* for item in self.cart\_items:
* # Calculate the cost of the current item (price \* quantity)
* item\_cost = item.item\_price \* item.item\_quantity
* # Add the cost of the current item to the total cost
* total\_cost += item\_cost
* # Return the total cost of the cart
* return total\_cost
  + - **print\_total**: Prints the total cost of the cart. If the cart is empty, it indicates that the cart is empty.
* def print\_total(self):
* if len(self.cart\_items) == 0:
* print(f"{self.customer\_name}'s Shopping Cart - {self.current\_date}")
* 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()}")
* total\_cost = 0
* for item in self.cart\_items:
* total\_cost += item.item\_price \* item.item\_quantity
* item.print\_item\_cost()
* print(f"Total: ${total\_cost}")
  + - **print\_descriptions**: Prints descriptions of each item in the cart.
* def print\_descriptions(self):
* print(f"{self.customer\_name}'s Shopping Cart - {self.current\_date}")
* print("\nItem Descriptions")
* for item in self.cart\_items:
* print(f"{item.item\_name}")

**User Menu and Functionality –**

Here is the menu of options to interact with the shopping cart:

* + **Add item to cart (a)**: Prompts the user for item details (name, price, quantity) and adds the item to the cart.
  + **Remove item from cart (r)**: Prompts the user for the name of the item to remove and attempts to remove it from the cart.
  + **Change item quantity (c)**: Placeholder for functionality to change the quantity of an item in the cart.
  + **Output items' descriptions (i)**: Displays the names of all items in the cart.
  + **Output shopping cart (o)**: Displays the current total cost of the cart along with the cost of each item.
  + **Quit (q)**: Exits the menu loop and ends the program.

**Summary –**

Working on the Online Shopping cart portfolio project was very helpful. I learned many new things in Python. Here is the summary of my key learnings.

1. Method Implementations – This was the main part of this assignment. We had to implement nine methods. The instructions in the assignment was very clear with each steps well defined. In real world I see lot more ambiguity in requirements and as a developer I have to clarify reqirements