**Module 8: Portfolio Project**

Amal Pulikkiyil

Colorado State University Global

CSC500: Principles of Programming

Isaac Gang

November 5, 2023

**Part 1 Pseudocode**

# Define the ItemToPurchase class

class ItemToPurchase:

Initialize name, price, quantity, and description

Method print\_item\_cost:

Calculate cost as price \* quantity

Output name, quantity, price, and cost

# Create two ItemToPurchase objects

item1 = ItemToPurchase()

Output "Item 1"

Input item1.name

Input item1.price

Input item1.quantity

item2 = ItemToPurchase()

Output "Item 2"

Input item2.name

Input item2.price

Input item2.quantity

# Calculate and output the total cost

Output "TOTAL COST"

Call item1.print\_item\_cost()

Call item2.print\_item\_cost()

total\_cost = item1.cost + item2.cost

Output "Total:", total\_cost

# Define the ShoppingCart class

class ShoppingCart:

Initialize customer\_name, current\_date, and cart\_items as an empty list

Method add\_item(added\_item):

Append added\_item to cart\_items

Method remove\_item(removed\_item):

For each item in cart\_items:

If item's name matches removed\_item's name:

Remove item from cart\_items

Return

Output "Item not found in cart. Nothing removed."

Method modify\_item(modified\_item):

For each item in cart\_items:

If item's name matches modified\_item's name:

If modified\_item's description is not "none", update item's description

If modified\_item's price is not 0.0, update item's price

If modified\_item's quantity is not 0.0, update item's quantity

Return

Output "Item not found in cart. Nothing modified."

Method get\_num\_items\_in\_cart:

Calculate and return the total quantity of items in cart\_items

Method get\_cost\_of\_cart:

Calculate and return the total cost of items in cart\_items

Method print\_total:

If cart\_items is empty, output "SHOPPING CART IS EMPTY"

Output customer\_name, current\_date, and the number of items in the cart

For each item in cart\_items, call item.print\_item\_cost()

Output "Total:", total cost of the cart

Method print\_descriptions:

If cart\_items is empty, output "SHOPPING CART IS EMPTY"

Output customer\_name, current\_date, and "Item Descriptions"

For each item in cart\_items, output item's name and description

# Define the print\_menu function

Function print\_menu(shopping\_cart):

Repeat until the user chooses to quit:

Output menu options:

Input choice

If choice is 'a':

Create an ItemToPurchase object

Output "ADD ITEM TO CART"

Input item.name, item.description, item.price, item.quantity

Call shopping\_cart.add\_item(item)

If choice is 'r':

Create an ItemToPurchase object

Output "REMOVE ITEM FROM CART"

Input removed\_item.name

Call shopping\_cart.remove\_item(removed\_item)

If choice is 'c':

Create an ItemToPurchase object

Output "CHANGE ITEM QUANTITY"

Input modified\_item.name, modified\_item.price, modified\_item.quantity, modified\_item.description

Call shopping\_cart.modify\_item(modified\_item)

If choice is 'i':

Output "OUTPUT ITEMS' DESCRIPTIONS"

Call shopping\_cart.print\_descriptions()

If choice is 'o':

Output "OUTPUT SHOPPING CART"

Call shopping\_cart.print\_total()

If choice is 'q':

Break

Else:

Output "Invalid choice. Please try again."

# Define the main function

Function main():

Input customer\_name

Input current\_date

Output "Customer name:", customer\_name

Output "Today's date:", current\_date

Create a ShoppingCart object with customer\_name and current\_date

Call print\_menu(shopping\_cart)

If the script is executed as the main program:

Call main()

**All code**

# -\*- coding: utf-8 -\*-

"""

Created on Sun Oct 8 11:02:50 2023

@author: pulik

"""

#%%

"""

Step 1: Build the ItemToPurchase class with attributes name, price, quantity,

a default constructor, and a pethod to print the item's cost

"""

class ItemToPurchase:

def \_\_init\_\_(self,

name = "none",

price = 0.0,

quantity = 0.0,

description = "none"):

self.name = name

self.price = price

self.quantity = quantity

self.description = description

def print\_item\_cost(self):

self.cost = self.price \* self.quantity

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

#%%

"""

Step 2: Prompt the user for two items and create two objects of the

ItemToPurchase class

"""

item1 = ItemToPurchase()

print("Item 1")

item1.name = input("Enter the item name:\n")

item1.price = float(input("Enter the item price:\n"))

item1.quantity = int(input("Enter the item quantity:\n"))

item2 = ItemToPurchase()

print("Item 2")

item2.name = input("Enter the item name:\n")

item2.price = float(input("Enter the item price:\n"))

item2.quantity = int(input("Enter the item quantity:\n"))

#%%

# Step 3: Add the costs of the two items together and output the total cost.

print("\nTOTAL COST")

item1.print\_item\_cost()

item2.print\_item\_cost()

total\_cost = item1.cost + item2.cost

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

#%%

"""

Step 4: Build the ShoppingCart class with attributes customer\_name, current\_date,

and cart\_items, as well as with methods add\_item(), remove\_item(), modify\_item(),

get\_num\_items\_in\_cart(), get\_cost\_of\_cart(), print\_total(), and print\_descriptions()

"""

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, added\_item):

self.cart\_items.append(added\_item)

def remove\_item(self, removed\_item):

for item in self.cart\_items:

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

self.cart\_items.remove(item)

return

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

def modify\_item(self, modified\_item):

for item in self.cart\_items:

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

if modified\_item.description != 'none':

item.description = modified\_item.description

if modified\_item.price != 0.0:

item.price = modified\_item.price

if modified\_item.quantity != 0.0:

item.quantity = modified\_item.quantity

return

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 len(self.cart\_items) == 0:

print("SHOPPING CART IS EMPTY")

return

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

print("Number of Items:", self.get\_num\_items\_in\_cart())

for item in self.cart\_items:

item.print\_item\_cost()

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

def print\_descriptions(self):

if len(self.cart\_items) == 0:

print("SHOPPING CART IS EMPTY")

return

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

print('Item Descriptions')

for item in self.cart\_items:

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

#%%

"""

Step 5: Implement the print\_menu() function that takes a ShoppingCart parameter

and outputs a menu of options to manipulate.

"""

def print\_menu(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")

choice = input('Choose an option: ').lower()

if choice == 'a':

item = ItemToPurchase()

print("\nADD ITEM TO CART")

item.name = input("Enter the item name:\n")

item.description = input("Enter the item description:\n")

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

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

ShoppingCart.add\_item(item)

elif choice == 'r':

removed\_item = ItemToPurchase()

print("\nREMOVE ITEM FROM CART")

removed\_item.name = input('Enter the name of the item to remove:\n')

ShoppingCart.remove\_item(removed\_item)

elif choice == 'c':

modified\_item = ItemToPurchase()

print("\nCHANGE ITEM QUANTITY")

modified\_item.name = input("Enter the name of the item to modify:\n")

modified\_item.price = float(input("Enter the new price:\n"))

modified\_item.quantity = int(input("Enter the new quantity:\n"))

modified\_item.description = input("Enter the new description:\n")

ShoppingCart.modify\_item(modified\_item)

elif choice == 'i':

print("\nOUTPUT ITEMS' DESCRIPTIONS")

ShoppingCart.print\_descriptions()

elif choice == 'o':

print('\nOUTPUT SHOPPING CART')

ShoppingCart.print\_total()

elif choice == 'q':

break

else:

print('Invalid choice. Please try again.')

#%%

def main():

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

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

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

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

shopping\_cart = ShoppingCart(customer\_name, current\_date)

print\_menu(shopping\_cart)

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

main()

**Result**

**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

**Git Repository**

<https://github.com/amalicious190810/Pulikkiyil_CSC500_Module7_CT>

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated