Sami Case

CSC-500 Principles of Programming

Module 6 Portfolio Milestone

**Source Code**

class ItemToPurchase:

def \_\_init\_\_(self, name = 'none', price = 0.00,quantity = 0, description=''):

self.name = name

self.price = price

self.quantity = quantity

self.description = description

def print\_item\_cost(self):

print('{:<20} {:.0f} @ ${:.2f} = ${:>6.2f}'.format(self.name,self.quantity,self.price, self.quantity\*self.price))

class ShoppingCart:

def \_\_init\_\_(self,customer\_name = 'none', current\_date = 'January 1, 2020',cart\_items=None):

if cart\_items is None:

cart\_items=[]

self.customer\_name = customer\_name

self.current\_date = current\_date

self.cart\_items = cart\_items

def add\_item(self):

item\_name = input('Enter item name: ')

item\_description = input('Enter item description: ')

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

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

new\_item = ItemToPurchase(item\_name,item\_price,item\_quantity,item\_description)

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

def remove\_item(self):

item\_name = input('Enter item name to remove: ')

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):

#modifies an item's description?, price, and/or quanitity

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

found = False

for item in self.cart\_items:

if item\_name == item.name:

found = True

print('Modifying {}'.format(item\_name))

new\_description = input('Enter new description for {} (current: {}) or press Enter to Keep'.format(item.name,item.description))

new\_price = input('Enter new price for {} (current: ${:.2f}) or press Enter to keep: '.format(item.name,item.price))

new\_quantity = input('Enter new quantity for {} (current: {}) or press Enter to keep: '.format(item.name,item.quantity))

if new\_description:

item.description = str(new\_description)

if new\_price:

item.price = float(new\_price)

if new\_quantity:

item.quantity = int(new\_quantity)

break

if not found:

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

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

count = 0

for item in self.cart\_items:

count += item.quantity

print('Number of items: {}'.format(count))

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

total\_cost = 0

for item in self.cart\_items:

total\_cost += item.quantity \* item.price

print('Total: ${:.2f}'.format(total\_cost))

def print\_total(self):

if not self.cart\_items:

print('SHOPPING CART IS EMPTY')

else:

for item in self.cart\_items:

item\_cost = item.quantity \* item.price

print('{} {} @ ${} = ${}'.format(item.name,item.quantity,item.price,item\_cost))

def print\_descriptions(self):

print('Item Descriptions')

for item in self.cart\_items:

print('{}: {}'.format(item.name,item.description))

def print\_menu(cart):

menu = (

'\nMENU\n'

'a - Add item to cart\n'

'r - Remove item from cart\n'

'c - Change item quantity\n'

'i - Output item descriptions\n'

'o - Output shopping cart\n'

'q - Quit\n'

)

command = ''

while command != 'q':

print(menu)

command = input('Choose an option: ').strip().lower()

if command == 'a':

cart.add\_item()

elif command == 'r':

cart.remove\_item()

elif command == 'c':

cart.modify\_item()

elif command == 'i':

cart.print\_descriptions()

elif command == 'o':

print('\n{}\'s Shopping Cart - {}'.format(cart.customer\_name,cart.current\_date))

cart.get\_num\_items\_in\_cart()

cart.print\_total()

cart.get\_cost\_of\_cart()

elif command == 'q':

print('Goodbye!')

else:

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

def main():

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

current\_date = input('Enter today\'s date: (e.g., January 1, 2024) ')

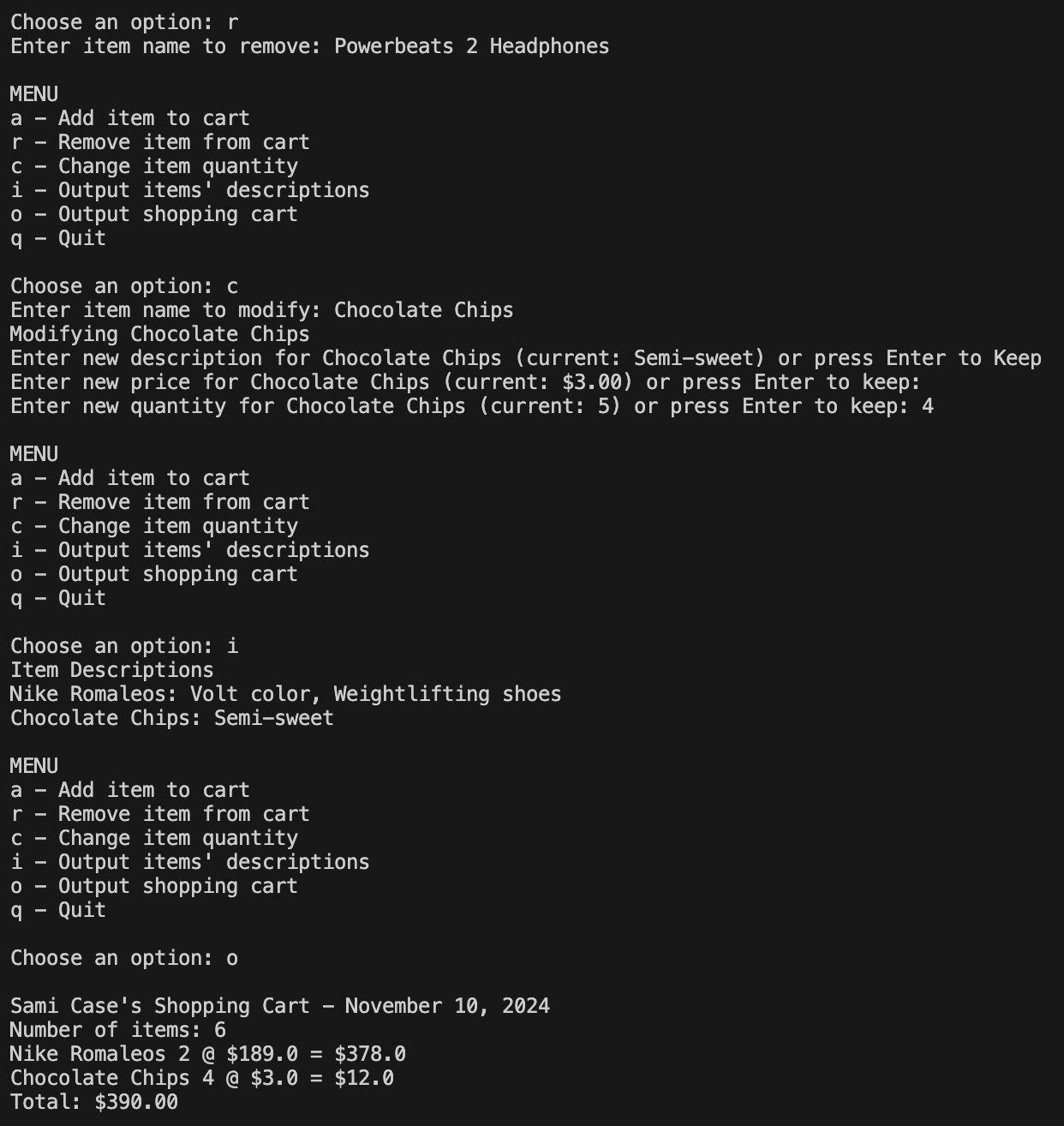
cart = ShoppingCart(customer\_name,current\_date)

print\_menu(cart)

main()

**Image of Successful Execution**



****

**GIT Repository**

<https://github.com/samilcase/CSC-500-Assignments>

A screenshot of a computer

Description automatically generated