**HOMEWORK PROJECT #1 – SHOPPING CART**

## **Part I.**

## Create two files to submit:

## ItemToPurchase.java – Class Definition

## ShoppingCartPrinter.java – Contains main() method

1. Build the *ItemToPurchase* class with the following specifications:

|  |  |
| --- | --- |
| Specifications | Description |
| ItemToPurchase(itemName) | itemName – The name will be a String datatype and Initialized in default constructor to “none”. |
| ItemToPurchase(itemPrice) | itemPrice – The price will be integer datatype and Initialized in default constructor to 0. |
| ItemToPurchase(itemQuantity) | itemQuantity – The quantity will be integer datatype Initialized in default constructor to 0. |
| ItemTax (category) | category - The category is a enum class of constants N, X, T  Each of them denote the following:  N - No tax for item  X - 8% tax for item  T - 10% tax for item |
| Public Member methods | Use of Public member methods (mutators & accessors)  setName() & getName()  setPrice() & getPrice()  setQuantity() & getQuantity()  setTaxCategory() & getTaxCategory() |

1. In main method of the driver (*ShoppingCartPrinter.java*) prompt the user for two items and create two objects of the *ItemToPurchase* class.

Your output should look like this:

Item 1

Enter the item name:

Chocolate Chips

Enter the item price:

3

Enter the item quantity:

1

Enter the tax category:

X

Item 2

Enter the item name:

Bottled Water

Enter the item price:

1

Enter the item quantity:

10

Enter the tax category:

T

1. Add the costs of the two items together, apply tax according to the conditions described and output the total cost.

Output Example:

TOTAL COST

Chocolate Chips 1 @ $3 = $3

Bottled Water 10 @ $1 = $10

Tax = $1.24

Total: $14.24

**Part 2. (5 points)**

This program extends the earlier "Online shopping cart" program. (Consider first saving your earlier program).

1. Extend the *ItemToPurchase* class per the following specifications

Private fields

* string itemDescription - Initialized in default constructor to "none"
* Parameterized constructor to assign item name, item description, item price, and itemquantity (default values of 0).

Public instance member methods

* setDescription() mutator & getDescription() accessor (2 pts)
* printItemCost() - Outputs the item name followed by the quantity, price, and subtotal
* printItemDescription() - Outputs the item name and description

Ex. of printItemCost() output:

Bottled Water 10 @ $1 = $10

Ex. of printItemDescription() output:

Bottled Water: Deer Park, 12 oz.

1. Create two new files

* ShoppingCart.java - Class definition
* ShoppingCartManager.java - Contains main() method

Build the *ShoppingCart* class with the following specifications. Note: Some can be method stubs (empty methods) initially, to be completed in later steps.

Private fields

* String customerName - Initialized in default constructor to "none"
* String currentDate - Initialized in default constructor to"January 1, 2020"
* ArrayList cartItems – where each item is of type ItemToPurchase
* Coupon code - Enum class with constants 10OFF, 20OFF, 30OFF

Public member methods

* Default constructor
* Parameterized constructor which takes the customer name and date as parameters
  + getCustomerName() accessor
  + getDate() accessor

addItem()

Adds an item to cartItems array. Has parameter ItemToPurchase. Does not return 

anything

removeItem()

* Removes item from cartItems array. Has a string (an item's name) parameter. Does not return anything
* If item name cannot be found, output this message: **Item not found in cart. Nothing removed.**

modifyItem()

* + Modifies an item's description, price, and/or quantity. Has parameter ItemToPurchase. Does not return anything.
  + If item can be found (by name) in cart, check if parameter has default values for description, price, and quantity. If not, modify item in cart.
  + If item cannot be found (by name) in cart, output this message: Item not found in cart. Nothing modified.

getNumItemsInCart()

* Returns quantity of all items in cart. Has no parameters.

getCostOfCart()

* Determines and returns the total cost of items in cart. Has no parameters.

applyCouponCode()

* A coupon code can be applied based on the following conditions:
  + 10OFF - $10 off the total price if current total exceeds $50
  + 20OFF - $20 off the total price if current total exceeds $100
  + 30OFF - $20 off the total price if current total exceeds $150

printTotal()

* + outputs total of objects in cart. Apply any applicable coupon codes.
  + If invalid coupon code is applied, output the message: INVALID CODE
  + If cart is empty, output this message: SHOPPING CART IS EMPTY

Example of PrintTotal()output:

John Doe's Shopping Cart - February 1, 2020

Number of Items: 8

Nike Romaleos 2 @$189 = $378

Chocolate Chips 5 @ $3 = $15

Powerbeats 2 Headphones 1 @ $128 = $128

Discount (30OFF): -$30

Total: $491

printDescription()

* + Outputs each item’s description.

Example of printDescription()output:

John Doe's Shopping Cart - February 1, 2020

Item Descriptions

Nike Romaleos: Volt color, Weightlifting shoes Chocolate Chips: Semi-sweet

Powerbeats 2 Headphones: Bluetooth headphones

1. In the file ShoppingCartManager.java, have the main method prompt the user for a customer's name and today's date.

* Output the name and date.
* Create an object of type ShoppingCart.

Output Example:

Enter Customer's Name: John Doe

Enter Today's Date:

February 1, 2020

Customer Name: John Doe

Today's Date: February 1, 2020

1. Implement the *printMenu()* method in ShoppingCartManager.java. *printMenu()* has a *ShoppingCart* parameter, and outputs a menu of options to manipulate the shopping cart. Each option is represented by a single character. Build and output the menu within the method
   * If there is an invalid character is entered, continue to prompt for a valid choice. Hint: Implement Quit before implementing other options.
   * Call *printMenu()* in the main() method.
   * Continue to execute the menu until the user enters q to Quit.

Output Example:

MENU

a - Add item to cart

d - Remove item from cart

c - Change item quantity

i - Output items' descriptions

o - Output shopping cart

p - Apply coupon code

q - Quit

Choose an option:

1. Implement Output shopping cart menu option

Output Example:

OUTPUT SHOPPING CART

John Doe's Shopping Cart - February 1, 2020

Number of Items: 8

Nike Romaleos 2 @ $189 = $378

Chocolate Chips 5 @ $3 = $15

Powerbeats 2 Headphones 1 @ $128 = $128

Discount (30OFF): -$30

Total: $491

1. Implement Output item's description menu option.

Output Example:

OUTPUT ITEMS' DESCRIPTIONS

John Doe's Shopping Cart - February 1, 2020

Item Descriptions

Nike Romaleos: Volt color, Weightlifting shoes

Chocolate Chips: Semi-sweet

Powerbeats 2 Headphones: Bluetooth headphones

1. Implement Add item to cart menu option.

Output Example:

ADD ITEM TO CART

Enter the item name:

Nike Romaleos

Enter the item description: Volt color, Weightlifting shoes

Enter the item price:

189

Enter the item quantity: 2

1. Implement Remove item menu option

Output Example:

REMOVE ITEM FROM CART

Enter name of item to remove: Chocolate Chips

1. Implement Change item quantity menu option.

Output Example:

CHANGE ITEM QUANTITY

Enter the item name: Nike Romaleos

Enter the new quantity: 3

1. Implement the Apply coupon code menu option. When an invalid coupon code or a coupon code which doesn’t meet the condition is applied, it should be rejected with the message ‘INVALID CODE’. Else, display the message ‘COUPON APPLIED’.

Output Example:

APPLY COUPON CODE

Enter coupon code: 20OFF

INVALID CODE

Enter coupon code: 10OFF

COUPON APPLIED

*Hint: Make new ItemToPurchase object and use ItemToPurchase modifiers before using modifyItem() method*

Rubric: 1 point for correct implementation of ItemToPurchase class

3 points for correct implementation of ShoppingCart class (0.5 deducted for each method not implemented or not working)

1 point for correct implementation of ShoppingCartManager

0.5 Points will be deducted each for lack of documentation, modularity and style