

OOP CA2 Report

Jack Levins



December 18, 2020

C19335843 Jack levins

C19335843@mytudublin.ie

Table of Contents

Contents

[Project Classes and Methods 2](#_Toc59184527)

[Customer Master class: 2](#_Toc59184528)

[**Loyal Customer Class:** 2](#_Toc59184529)

[**Bargain Hunter Class:** 3](#_Toc59184530)

[Shopping Cart Class 4](#_Toc59184531)

[Product Class 5](#_Toc59184532)

[User Manual 6](#_Toc59184533)

[Difficulties and Challenges 7](#_Toc59184534)

[References/Bibliography 8](#_Toc59184535)

# Project Classes and Methods

## Customer Master class:

**Description:** This master class is used to form a basis of the subclasses using only their essential details

**Methods:**

init: The init methods takes 4 arguments:

A customer’s name, address, accountID and their preferred currency

The default values are empty strings for name and address, 0 for accountID and euro for preferred currency.

str: The string method prints out the attributes mentioned above in a tidy way with their values given.

### **Loyal Customer Class:**

**Description:** This class is identical to the Customer masterclass except the exclusive products element is added in both the initialisation and the string. It takes the same attributes but also allows the customer to shop for exclusive items.

**Methods:**

**init:** The init methods takes 4 arguments:

A customer’s name, address, accountID and their preferred currency, the same as the Customer masterclass.

The default values are empty strings for name and address, 0 for accountID and euro for preferred currency.

An attribute of exclusive is added to indicate exclusive products can be bought and seen by this customer.

**str:** The string method uses the customer masterclass’ str method and adds on a string of exclusive product access being allowed.

### **Bargain Hunter Class:**

**Description:** This class is the same as the customer master class, exclusive products cannot be shopped for as the customer is not considered a loyal customer and does not have this access so their shopping experience will differ in this way.

**Methods:**

**init:** The init methods takes 4 arguments:

A customer’s name, address, accountID and their preferred currency, the same as the Customer masterclass.

The default values are empty strings for name and address, 0 for accountID and euro for preferred currency.

An attribute of exclusive is added to indicate exclusive products can be bought and seen by this customer. This will be set to false.

**str:** The string method uses the customer masterclass’ str method and adds on a string of exclusive product access being disallowed.

## Shopping Cart Class

**Description:** The shopping cart class is responsible for the majority of user tasks including adding items to basket, storing them, removing items from basket

**Methods:**

**init:** The customer is passed as an argument and their exclusive access and viewable products are considered when items are being added or removed to/from their basket. Empty list and dictionary will be initialised in order to facilitate the functions used in this class and the amount will be set to 0 to start off.

**str:** This method will display the items in the basket followed by the basket total and the customer’s currency will be used to display the total amount

**get\_amount:** This returns the amount in the basket (subtotal)

**get\_customer:** This returns the customer that is using the shopping cart

**add\_item:** This will add an item to the basket dictionary if it isn’t already in it and make the quantity 1 and add the product to the basket list, if it is in the basket list the quantity will increase by 1 and the amount of the item will be added to the subtotal.

**remove\_item:** This will remove an item from the dicionary the basket is stored in if its quantity will be 0 and removed from the list of items currently in basket, if the quantity will still be 1 or more its quantity in the basket dictionary will be decreased by 1 and still remain in the products in basket list. The product amount will be deducted from the subtotal.

**Note:** The overloaded operators ‘+’ and ‘-‘ will be used for the add and remove items from basket using a function call.

**get\_basket\_quantity:** This will return the total amount of items in the basket

**checkout:** This will take all items from basket get the amount, the items in basket and ask the user for a delivery option: priority or standard, shipping fees are then added, and it will ask the user to confirm the purchase if it is confirmed a message will appear and the program will stop after.

If it is not confirmed the shipping fee is removed from the subtotal and the program continues to main after a message is printed.

## Product Class

**Description:** The product class is used to initialise products and show them based on the customer using it. This includes their preferred currency and whether or not they have access to exclusive products or not.

**Methods:**

**init:** The init method takes a customer as argument, from the customer their preferred currency is taken and also if they have access to exclusive products. An empty dictionary is formed. The key is the product name and the value is the price. A text file containing these is opened and parsed into the dictionary. The amount in the file is based in euros but the convert\_currency function can be called if it is a different currency and the products dictionary will reflect this.

If exclusive products are allowed, the process repeats with a different file containing the exclusive products in the same format and this dictionary is merged to the original using the update method to have access to all products.

**str:** The string method prints the product name and then their amount line by line based on what rights the customer has.

**convert\_currency:** The convert\_currency function will convert the amount given from euros to the customer’s preferred currency, either dollars or pounds. This makes the customer experience much easier to follow.

# User Manual

When greeted with the command line menu, the customer must create a customer using the various options given. Option 1 must be chosen first unless the user wishes to quit. The user must then enter their details as asked for by the command line.

When the customer creates successfully, they will return to the main menu, 1 becomes an invalid option as they have already created a customer. The user can then add products to basket. To see what products are available first they must enter 2 on the main menu. Products will be then displayed with the preferred currency after conversion.

If a user wishes to add or remove an item to/from their basket they should enter 3 on the main menu, when prompted they should enter the + character or type add and enter, for remove they should use the – symbol or type remove and enter. They should then enter the product they wish to add/remove, the program will take care of any errors such as removing a item that isn’t already in your basket and return to the main menu, otherwise the item will be added or removed.

To confirm this, enter 4 on the main menu. This will display your current cart and it’s subtotal. If the price seems to be too high the quantity of one or more items may be more than one and will be reflected in the price.

If the customer is satisfied and wants to check out, they will enter checkout when 5 is entered. As long as one or more items are in the basket. A delivery option is presented, priority or standard, choose an option, enter 1 for priority (next-day delivery) or enter 2 (3-5 days) then, a confirmation will appear, and the user should enter y if they wish to proceed to buy or n to cancel. If checkout is confirmed a message will be printed to the user before closing. If checkout is cancelled a message is displayed to the user and then the program returns to the main menu with no changes.

# Difficulties and Challenges

Implementing error checking took a lot of careful thought and testing to get right. This did take more time than intended but the results were much better than anticipated. A lot of this error checking was down to incorrect user input. Other errors included what happens if a condition is not as expected in terms of shopping itself. Such as entering checkout without an item in the cart or trying to use features without creating an account. This all had to be handled carefully in the way I thought was most appropriate.

The creative element took a lot of different ideas and possible solutions. As ideas emerged, I had to think about how difficult it would be to implement in the code and make sure I was not using too much valuable time on one piece of the puzzle. In the end I went for shipping, which is quite basic in a sense as it’s just a rate added to the total at the end, however through personal experience with using Amazon I implemented different options. The options included priority delivery as well as standard. To make that a bit more complex I added in quantity check, so if total quantity exceeded 4 delivery increased by a fixed amount.

In my experience adding overload operator took more creativity than the creative element itself. Thinking of what could be implemented in a shopping environment was very difficult as standard operations seemed sufficient for the program however, I ended up choosing the add and sub operators. To do this I used my already made functions “add\_item” and “remove\_item” and thought of these as adding and subbing something to and from the basket. I could’ve renamed these functions and just used them but I instead felt that both made sense so I made and add and sub as a function that calls the original function, so they work the same way. This was due to how late in the project I had thought of this

Implementing str methods required thought in what was most important to display to the user when required. As seen in my shopping cart str method you will notice a lot of overlapping attributes, this is because of the amount of data required in a basket such as total amount, quantity of each product, price of each product and much more and implementing the str method required a clearer way of displaying the current cart which is why I used the list as opposed to the dictionary to keep it concise enough.

Other than these issues there were not many major problems, some things changed during the coding of the programming and caused a lot of shifting code and fixing whatever that particular piece of code which caused half the program being changed at a time of change.

Time management was difficult, a lot more than the last CA with many obstacles encountered, many other CA and assignment/tests due all at once as well as personal circumstances so thoughtful planning was needed.

# References/Bibliography

No external material used for this report.