

OBJECT ORIENTED SOFTWARE DEVELOPMENT

Final Assignment TU259

Bearach Byrne
C15379616

School of Computer Science
TU Dublin – City Campus

09/05/2021

Table of Contents

Table of Contents.....	2
Table of Figures	3
Declaration	4
1. Introduction	5
1.1. Application Design Requirements	5
2. Additional Features	6
2.1. Discount for loyal customers	6
2.2. Product Class	6
2.3. Lambda Function for sorting of Dictionary	6
2.4. External Text File for Products	6
3. Classes & Methods	8
3.1. ShoppingCart	8
3.1.1. Attributes	8
3.1.2. Methods	8
3.2. Customer	8
3.2.1. LoyalCustomer	8
3.2.2. BargainHunter	8
3.3. Product	8
4. User Manual	9
4.1. Inputs	9
4.2. Main Menu	9
4.3. Create Account	9
4.4. List Available Products	10
4.5. Add/Remove Products to/from the Cart	11
4.6. See Current Cart	11
4.7. Checkout	12
4.7.1. Bargain Hunter Checkout	12
4.7.2. Loyal Customer Checkout More than €200	12
4.7.1. Loyal Customer Checkout Less Than €200	13
4.8. Quit	13
5. Difficulties & Challenges	14
5.1. Object Attributes	14
5.2. Product Dictionary	15
5.3. The Main Menu Function	15
5.4. Length of the Program	15
5.5. Formatting of Data Tables	15
5.6. Dealing With Classes	16
5.7. Class Attributes	16

Table of Figures

Figure 1 - Loyal Customer Checkout More than €200	6
Figure 2 - Loyal Customer Checkout Less than €200	6
Figure 3 - The lambda function used to sort the dictionaries. This method was taken from https://www.geeksforgeeks.org/ways-sort-list-dictionaries-values-python-using-lambda-function/	6
Figure 4 - the item_list text file.....	7
Figure 5 - The product list for a loyal customer.....	10
Figure 6 - The product list for a bargain hunter.....	11
Figure 7 - Bargain hunter attempting to add an exclusive item to their cart	11
Figure 8 - Showing the user's cart	12
Figure 9 - Checkout menu.....	12
Figure 10 - Successful checkout.....	12
Figure 11 - Loyal Customer Checkout More than €200	13
Figure 12 - Loyal Customer Checkout Less than €200	13
Figure 13 - Program Quit	13
Figure 14 - The original implementation that gave an error	14
Figure 15 - The error message given	14
Figure 16 – Modifying the return of the populate_loyal_dict so that it also outputs a simple dictionary that is not related to a class.....	14
Figure 17 - The data table showing the customer information	15

Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

Bearach Byrne

Bearach Byrne

09/05/2021

1. Introduction

This section will provide an introduction to the assignment, outlining the design requirements along with a short summary of the assignment brief.

1.1.Application Design Requirements

The program that was created for this assignment had to cover a number of design requirements outlined in the brief.

- It must use at least 2 classes, ShoppingCart and Customer, with the Customer Class containing 2 further subclasses LoyalCustomer and BargainHunter.
- Each class should have an `__init__` and `__str__` method.
- There should be a command line menu with 6 options:
 - 1. Create a Customer
 - 2. List available products
 - 3. Add/Remove a product to the shopping cart
 - 4. See current shopping cart
 - 5. Checkout
 - 6. Quit
- Error Checking
- Function Annotations
- Composition, Aggregation/Inheritance
- Correct usage of Data Structures
- Private & Public Attributes

2. Additional Features

2.1. Discount for loyal customers

One of the additional features added to this is the discount provided to loyal customers if they spend over €200. If they do not have €200 worth of items in their cart they will be told how much they need to add to avail of this.

```
-----
You have chosen to checkout

Items in cart:
Item : Quantity
pen  : 1
pc   : 1
Your subtotal is: €700.99

-----
Because of your loyalty status, you also get a 5% discount on this purchase as it is over €200
This brings your total to: €665.94
Do you want to proceed with checkout? (Y/N):
```

Figure 1 - Loyal Customer Checkout More than €200

```
-----
You have chosen to checkout

Items in cart:
Item : Quantity
pen  : 5
Your subtotal is: €5.0

-----
Because of your loyalty status, you are eligible for a 5% discount on purchases over €200
Add a further € 195.0 worth of items to your cart to avail of this discount!
Do you want to proceed with checkout? (Y/N): |
```

Figure 2 - Loyal Customer Checkout Less than €200

2.2. Product Class

An additional product class was added to make dealing with the products involved easier. This was not fully utilised; I had planned more functionality/methods to be implemented before running out of time. For instance I was planning for a method to allow for prices of items to be changed within the program, which would then be written to the text file. The method still exists within the program however it is not used anywhere as the design of it was never finished.

2.3. Lambda Function for sorting of Dictionary

A lambda function was used in order to sort the dictionaries for the loyal customers and bargain hunters in different ways.

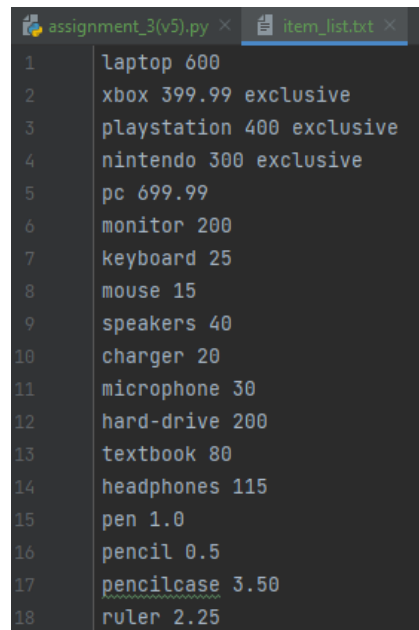
```
for elem in line_list:
    my_dict[elem[0]] = float(elem[1])
# Sorts the dict into descending price using a lambda function
sorted_dict = dict(sorted(my_dict.items(), key=lambda kv: kv[1], reverse=True))
# If the user is a bargain hunter, don't add the loyal exclusives to the list
```

Figure 3 - The lambda function used to sort the dictionaries. This method was taken from <https://www.geeksforgeeks.org/ways-sort-list-dictionaries-values-python-using-lambda-function/>

2.4. External Text File for Products

A simple text file was used to input the products available for purchase to the customers. It contains a list of the items for sale. Each line of this must be in a specific format in order to be read by the program correctly. Each line must consist of either 2 or 3 elements separated by a space. Element 1 being the item name, element 2

being the price of the item and the optional element 3 being exclusivity of the item, with exclusive items being marked 'exclusive' and non-exclusive items not having a 3rd element.

A screenshot of a text editor window with two tabs: 'assignment_3(v5).py' and 'item_list.txt'. The 'item_list.txt' tab is active, showing a list of 18 items. Each line consists of an item name, a price, and an optional 'exclusive' status. The items are: laptop (600), xbox (399.99, exclusive), playstation (400, exclusive), nintendo (300, exclusive), pc (699.99), monitor (200), keyboard (25), mouse (15), speakers (40), charger (20), microphone (30), hard-drive (200), textbook (80), headphones (115), pen (1.0), pencil (0.5), pencilcase (3.50), and ruler (2.25).

```
1 laptop 600
2 xbox 399.99 exclusive
3 playstation 400 exclusive
4 nintendo 300 exclusive
5 pc 699.99
6 monitor 200
7 keyboard 25
8 mouse 15
9 speakers 40
10 charger 20
11 microphone 30
12 hard-drive 200
13 textbook 80
14 headphones 115
15 pen 1.0
16 pencil 0.5
17 pencilcase 3.50
18 ruler 2.25
```

Figure 4 - the item_list text file

3. Classes & Methods

3.1.ShoppingCart

The shopping cart class only has 2 attributes, items which is a dictionary containing the user's cart, and loyal which denotes the loyalty of the customer.

3.1.1. Attributes

The shopping cart class only had 2 attributes, items which was the current cart contents and loyal which represents the loyalty status of the customer the cart is associated with.

3.1.2. Methods

The `__init__` and `__str__` methods deal with the initialising of and printing of the cart.

add_item – This method is used to add items to the user's cart from the product dictionary. The user can only add products from the catalogue that they are authorised to view (loyal customers can see exclusive items, bargain hunters cannot).

remove_item – This method is used to remove items from a user's cart. In order to remove an quantity of an item the user must have a greater quantity of that item in their cart.

list_cart – This function lists all the items that are currently in a user's cart line by line.

checkout – This method is used to calculate the total value of the items in the cart. If the user is a loyal customer, they will be given a discount on purchases over €200, and if the purchase is less than €200, they will be told how much they need to add to avail of this discount.

Once the user views their total, they are asked to confirm if they would like to checkout, if they answer no, the program brings them back to the main menu, if yes it prints a thank you message and ends the program.

3.2.Customer

The customer superclass has 6 attributes, ID, name, date of birth, email, phone, cart, and only 2 methods, `__init__` and `__str__`.

3.2.1. LoyalCustomer

The only additional attribute that the LoyalCustomer class has over the Customer superclass is the addition of a Loyal attribute which is always True for the class. Due to how my program is structured, my program would function exactly the same with a single customer class with the loyal attribute differentiating between loyal customers and bargain hunters. I did not end up using any additional attributes for these, however in future additional attributes could be things like loyalty points for loyal customers.

3.2.2. BargainHunter

The only additional attribute that the BargainHunter class has over the Customer superclass is the addition of a Loyal attribute which is always True for the class.

With some optimisation, the amount of code used to define the `__str__` methods for these classes could be reduced down to a minimum, however this was not a priority due to time constraints and as such is quite inefficient.

3.3.Product

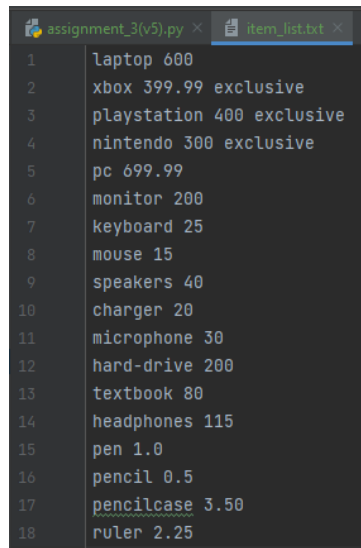
An additional product class was added to make dealing with the products involved easier. This was not fully utilised; I had planned more functionality/methods to be implemented before running out of time. For instance I was planning for a method to allow for prices of items to be changed within the program, which would then be written to the text file. The method still exists within the program however it is not used anywhere as the design of it was never finished.

4. User Manual

This program is a simple shopping application for a store.

4.1. Inputs

The main input to the application is the product list. This is done in the form of a simple text file (called item_list.txt), that contains a list of the items for sale. Each line of this must be in a specific format in order to be read by the program correctly. Each line must consist of either 2 or 3 elements separated by a space. Element 1 being the item name, element 2 being the price of the item and the optional element 3 being exclusivity of the item, with exclusive items being marked 'exclusive' and non-exclusive items not having a 3rd element.

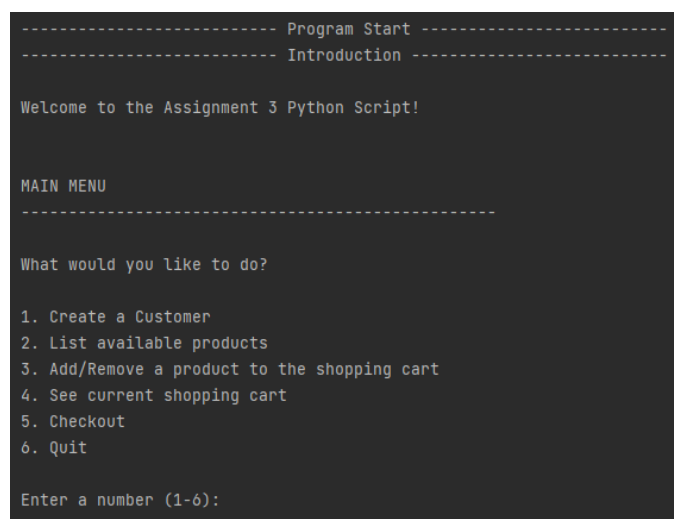


```
1 laptop 600
2 xbox 399.99 exclusive
3 playstation 400 exclusive
4 nintendo 300 exclusive
5 pc 699.99
6 monitor 200
7 keyboard 25
8 mouse 15
9 speakers 40
10 charger 20
11 microphone 30
12 hard-drive 200
13 textbook 80
14 headphones 115
15 pen 1.0
16 pencil 0.5
17 pencilcase 3.50
18 ruler 2.25
```

4.2. Main Menu

The main menu has 6 options for the user to choose from:

1. Create a Customer
2. List available products
3. Add/Remove a product to the shopping cart
4. See current shopping cart
5. Checkout
6. Quit



```
----- Program Start -----
----- Introduction -----

Welcome to the Assignment 3 Python Script!

MAIN MENU
-----

What would you like to do?

1. Create a Customer
2. List available products
3. Add/Remove a product to the shopping cart
4. See current shopping cart
5. Checkout
6. Quit

Enter a number (1-6):
```

A user must create a customer account before performing any account options (options 2-5).

4.3. Create Account

When a user chooses to create an account, they must enter personal information such as name and date of birth (DOB). Follow the on screen prompts to complete this process.

```

Enter a number (1-6): 1
-----

CREATE CUSTOMER SUBMENU
-----

We're going to need some information to create your account.

Enter your first name: bealach
Enter your last name: byrne
Enter your DOB in the format 'dd/mm/yyyy' : |

```

The information entered will be validated to ensure accurate information is supplied, the user will not be allowed to progress until information is entered in the correct format.

```

Enter your first name: bealach
Enter your last name: byrne
Enter your DOB in the format 'dd/mm/yyyy' : 6
Looks like you've entered an illegal input. Try entering a date in the format dd/mm/yyyy
Enter your DOB in the format 'dd/mm/yyyy' : |

```

Once all the personal information has been supplied, the user will be asked if they would like to join the store's loyalty programme. This is a subscription service that provides access to exclusive products and deals, along with a 5% discount on purchases above €200.

4.4. List Available Products

This option simply lists out all products available for purchase to the user. Loyal customers will be able to see all products sorted by price from high to low.

```

Enter a number (1-6): 2
-----

You have chosen to list the products
Product Number: 1

Product Name: pc
Product Price: €699.99
-----

Product Number: 2

Product Name: laptop
Product Price: €600.0
-----

Product Number: 3

Product Name: playstation
Product Price: €400.0
-----

```

Figure 5 - The product list for a loyal customer

Bargain Hunters will not be able to see the exclusive products, and instead will have the available products listed from low price to high price.

```

Enter a number (1-6): 2
-----
You have chosen to list the products
Product Number: 1

Product Name: pencil
Product Price: €0.5
-----

Product Number: 2

Product Name: pen
Product Price: €1.0
-----

Product Number: 3

Product Name: ruler
Product Price: €2.25
-----

Product Number: 4

Product Name: pencilcase
Product Price: €3.5
-----

```

Figure 6 - The product list for a bargain hunter

4.5.Add/Remove Products to/from the Cart

Option 3 brings up the add/remove submenu. Here, users can choose to add items to their cart from the product list or remove items already in their cart.

```

-----
You have chosen to add or remove an item from the cart.
1. To add item to cart
2. To remove item from cart
3. To exit to main menu
Enter your selection (1 - 3): |

```

Be careful to only add items that are available to the customer (bargain hunters cannot add exclusive items to their cart).

```

-----
You have chosen to add or remove an item from the cart.
1. To add item to cart
2. To remove item from cart
3. To exit to main menu
Enter your selection (1 - 3): 1
What item would you like to add?: xbox
How many?: 1
It looks like you entered an incorrect item name. Try viewing the product list again.

```

Figure 7 - Bargain hunter attempting to add an exclusive item to their cart

4.6.See Current Cart

Option 4 will display the current contents of the user's cart.

```
Enter a number (1-6): 4

-----

You have chosen to print your current cart

Items in cart:
Item   :   Quantity
pen    :         3
pc     :         4
laptop :         1
```

Figure 8 - Showing the user's cart

4.7.Checkout

Option 5 will bring the user to the checkout.

4.7.1. Bargain Hunter Checkout

Bargain hunters will simply be shown their cart and a subtotal, they are then asked if they wish to proceed. If they choose no at this menu, they will be brought back to the main menu where they can edit their cart. If they choose yes, they will get a thank you message, their cart will be emptied and the program will end.

```
Enter a number (1-6): 5

-----

You have chosen to checkout

Items in cart:
Item   :   Quantity
pen    :         3
pc     :         4
laptop :         1
Your subtotal is: €3402.96

-----

Do you want to proceed with checkout? (Y/N): |
```

Figure 9 - Checkout menu

```
-----

Do you want to proceed with checkout? (Y/N): y
Thank you for your purchase!

-----Thanks For Your Time-----
----- Come Back Anytime -----
----- Program End -----
-----
```

Figure 10 - Successful checkout

4.7.2. Loyal Customer Checkout More than €200

If a loyal customer chooses to checkout and they have more than €200 worth of items in their cart, a 5% discount will automatically be applied to their purchase.

```

-----
You have chosen to checkout

Items in cart:
Item   :   Quantity
pen    :       1
pc     :       1
Your subtotal is: €700.99

-----

Because of your loyalty status, you also get a 5% discount on this purchase as it is over €200
This brings your total to: €665.94
Do you want to proceed with checkout? (Y/N):

```

Figure 11 - Loyal Customer Checkout More than €200

4.7.1. Loyal Customer Checkout Less Than €200

If a loyal customer chooses to checkout and they have less than €200 worth of items in their cart, the system will tell them they can avail of a 5% discount if they spend over €200. It will tell them how much more they need to add to their cart to reach this.

```

-----
You have chosen to checkout

Items in cart:
Item   :   Quantity
pen    :       5
Your subtotal is: €5.0

-----

Because of your loyalty status, you are eligible for a 5% discount on purchases over €200
Add a further € 195.0 worth of items to your cart to avail of this discount!
Do you want to proceed with checkout? (Y/N): |

```

Figure 12 - Loyal Customer Checkout Less than €200

4.8. Quit

Option 6 allows the user to quit the program entirely. This can be done whether the user has created an account or not.

```

What would you like to do?

1. Create a Customer
2. List available products
3. Add/Remove a product to the shopping cart
4. See current shopping cart
5. Checkout
6. Quit

Enter a number (1-6): 6

-----Thanks For Your Time-----
----- Come Back Anytime -----
----- Program End -----
-----

```

Figure 13 - Program Quit

5. Difficulties & Challenges

There were many difficulties & challenges that came up during the development of this program.

5.1. Object Attributes

When writing the checkout method for the Shopping Cart class, I ran into a `TypeError` when trying to multiply the value of the product (taken from the product dictionary which is an attribute of the Product Class) by the quantity of the product (taken from the cart dictionary), however this gave me a `TypeError` that I was unable to fix.

As I did not have enough time to find a proper solution for this, the workaround that I used was to create a duplicate of the product dictionary that was a standalone dictionary in the program (named `product_dict_notobj`), as opposed to the product dictionary that is an attribute of the Product Class.

```
def checkout(self):
    subtotal = 0
    if bool(self.items) == True:
        for key in self.items:
            item_total = 0
            quant = int(self.items[key])
            price = product_dict[key]
            item_total = quant * price
            subtotal += item_total
            subtotal_str = "€"
            subtotal_str += str(subtotal)
        print("Your subtotal is:", subtotal_str)
        return subtotal_str
    else:
        print("There's nothing in your cart yet! Try adding something and coming back here!")
```

Figure 14 - The original implementation that gave an error

```
You have chosen to checkout
Traceback (most recent call last):
  File "C:\Users\Bearach\OneDrive - Technological University Dublin\1. TU259\1.
    main_menu()
  File "C:\Users\Bearach\OneDrive - Technological University Dublin\1. TU259\1.
    cust_cart.checkout()
  File "C:\Users\Bearach\OneDrive - Technological University Dublin\1. TU259\1.
    item_total = quant * price
TypeError: unsupported operand type(s) for *: 'int' and 'Product'
```

Figure 15 - The error message given

```
# sorts the dict into descending price using a lambda function
sorted_bargain_dict = dict(sorted(my_dict.items(), key=lambda kv: kv[1], reverse=False))
# Creates an instance of the product class for each product in the sorted dictionary
for item in sorted_bargain_dict:
    product_dict[item] = Product(item, my_dict[item])
my_file.close()
return product_dict, sorted_bargain_dict
```

Figure 16 – Modifying the return of the `populate_loyal_dict` so that it also outputs a simple dictionary that is not related to a class

5.2.Product Dictionary

This gave a lot of trouble and in the end, I am still not satisfied with how I dealt with it. In the end I had to re-set the populate_dict function as a variable (product_dict) in any method that required the full product dictionary. This way of doing seems to be quite inefficient, however due to the time pressure of the assignment once I had it working, I did not have time to find a better method of it.

5.3.The Main Menu Function

Given more time, this function would be further divided up into separate functions so that each menu option featured a single line calling whatever function/method was needed. As it stands the entire main menu function is very large and quite messy.

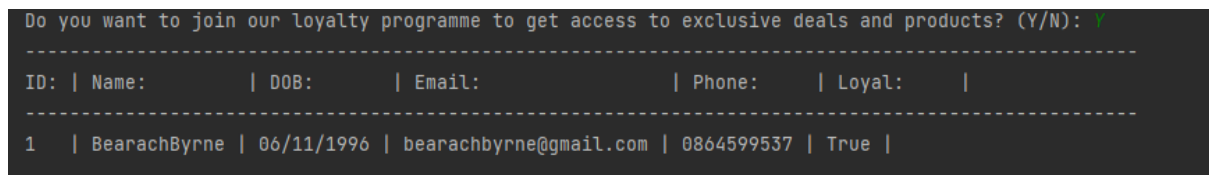
5.4.Length of the Program

One thing I struggled with when writing/updating this program was due to the sheer size of it, being hundreds of lines, it required constant restructuring so that everything was laid out in a coherent manner and could be easily modified/updated. Going forward I will definitely try to keep my programs laid out in a coherent manner as it makes editing much easier.

5.5.Formatting of Data Tables

I found formatting tables to be quite tricky for this project. I experimented with a few different layouts for the various data tables but never quite found one style that worked.

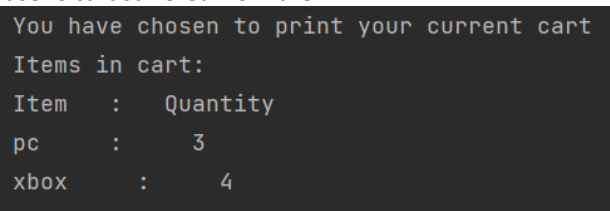
The table used to print customer information is extremely dependant on the length of the strings the user inputs, as the information headers are a constant width/spacing. So if a user with a very short name or email address creates an account the columns will misaligned. There would be two potential fixes for this, using a third party module to scale the dimensions of the columns based on the text or by printing it in a linear format line by line.



```
Do you want to join our loyalty programme to get access to exclusive deals and products? (Y/N): Y
-----
ID: | Name:          | DOB:          | Email:          | Phone:          | Loyal:          |
-----
1  | BearachByrne    | 06/11/1996   | bearachbyrne@gmail.com | 0864599537     | True           |
```

Figure 17 - The data table showing the customer information

The table used to print the user's cart suffered from the



```
You have chosen to print your current cart
Items in cart:
Item   :   Quantity
pc     :       3
xbox   :       4
```

```
-----  
You have chosen to list the products  
Product Number: 1  
  
Product Name: laptop  
Product Price: €600  
-----  
Product Number: 2  
  
Product Name: pc  
Product Price: €600  
-----  
Product Number: 3  
  
Product Name: xbox  
Product Price: €400  
-----  
Product Number: 4  
  
Product Name: playstation  
Product Price: €400  
-----  
Product Number: 5  
  
Product Name: nintendo  
Product Price: €300  
-----  
Product Number: 6  
  
Product Name: monitor  
Product Price: €200  
-----
```

5.6. Dealing With Classes

One of the most challenging parts of this assignment for me was learning how to design and implement classes as a concept. As such, when I was starting out with the assignment, I tended to create functions outside of the classes so that I could get to grips with the logical steps needed for each portion of the application. This ended up causing me some problems when trying to move the logic from functions in the main code into class methods as the naming schemes and how object/variables are called is slightly different. As such I think I created a lot more work for myself by doing it this way. However it did enable me to learn the basics of what was needed for the functionality of the program before implementing classes, which I did not have much experience with. I feel like I have learned a tremendous amount about not only classes, but how to structure larger and more complex python programs.

5.7. Class Attributes

In the beginning of this project, I had included far more attributes than are included in the final version of the program, however I found that I was including attributes that did not quite make sense for each class. As such I ended up simplifying the number of attributes down to the bare minimum, which made the classes a lot simpler but at the detriment of additional functionality. By the time I got around to finishing off the primary functionality of the program I then did not have enough time to go back and add in the attributes and the functionality that I wanted (functionality such as a customer credit system for loyal customers where they gain small amounts of credit or loyalty points which can be redeemed on future purchases).