Object Oriented Software Development

Final Assessment

Marks and Submission

This CA is worth **60%** of your overall module mark

Submission deadline: 09/05/2021, 11:59pm

Penalty for late submission: 4^d%, where d is the number of days. Note that after 3 days the penalty is set to 100%.

For the project you need to provide your **code** and a **report**. Any submission without a report will not be marked. More details of evaluation in the marking scheme.

Plagiarism

Plagiarism is a serious offence – do not use other people's code, as well as do not share your files with others and do not share them online (e.g. public github)!

If you include in your code content outside the class material, make sure to add the reference or link to the website. Adding extra features to the system and using self-learned material is encouraged, but it **must** be properly referenced.

Problem Description:

For this assignment you are asked to develop an application to manage a shopping cart, such as the usual ones found online.

Design a **shopping cart class** in Python. Think of what attributes and methods you have to include in your class and why. Use a **dictionary** to implement at least one attribute.

Design a **customer class**, with **two sub classes**, Loyal Customers and Bargain Hunters. Loyal Customers can buy exclusive products, while Bargain Hunters can buy any other products listed from low to high price.

Each class should have an **__init__** and **__str__** methods. For each __str__ method, think what information each class should provide when you print their instances. Include in one of your classes at least two methods available in Python to **overload operators**. Demonstrate the use of such operators with different instances of your class.

Implement a command line menu with the following options:

```
1. Create a customer.
2. List products.
3. Add/remove a product to the shopping cart.
4. See current shopping cart.
5. Checkout.
```

Option 1 has to be performed before listing or adding any products to the shopping cart. Once a customer is created he/she should be able to see the products available to him/her, to loop over the options to add/remove products, and to see the current products in the shopping cart. The checkout is available once at least one product is added to the shopping cart. To perform the checkout, confirm that the customer is happy with the current items in the cart and the amount due. If it is confirmed, print a thank you message and exit the program.

Finally, provide a **testing function** “def test():” in the main scope**,** which performs all the required tasks of this system without using the command line interface. The only purpose of this function is to be used by me to evaluate if you have instantiated all the classes and tested the system accordingly.

Hints and General Guidelines:

- Start with the easy parts. Define your constructors and add new functionalities one by one.
- Once a method has been coded test it before moving on to the next part
- Make sure you include any relevant **error checking** and handle unexpected input
- Make sure to use **function annotations**
- Once you have defined a class you can use its instances as any other type. Remember to use **composition, aggregation and/or inheritance** if applicable.
- Think about which data structure is more appropriate to the pieces of information you need to store: list, string, dictionary, tuple, etc.
- Think if each attribute needs to be **private or public**. Remember to add **get/set** methods for private attributes that are accessed outside the class.

- Add **docstrings** to all your classes and methods

Sample Marking Schemes

Correct functionality (50%)

- Code compiles and runs with no problem.
- All required classes have been coded.
- Required functionalities work properly.
- Inheritance and composition/aggregation applied.
- Ability to deal with exception handling.
- Functions/methods are used correctly to break down the problem. They are all used for a single purpose or for a single task.
- Nice use of data structures such as dictionaries and sets.
- Operator overloading used correctly.
- Command line menu is well implemented, and all its options work as expected
- Testing has been performed and there is enough code to instantiate and use all the options of the system in the testing function.

Creativity (20%) 

- Additional features that were not required and that fit well in the system have been included. For example:
    - Think what else a shopping cart system might have. Perhaps other types of customers, shipping options, payment in different currencies, etc.
    - Additional classes have been developed.
    - Use of other concepts you self-study outside the content delivered in the course.

Layout and use of comments (10%)
Indentation and white space have been used appropriately. Code is easy to read, and naming conventions have been adopted. Code is well documented, and it is easy to understand.

Report (20%, submissions without a report will not be marked)
Understanding of the code will be evaluated based on a report written by you explaining your system. It must contain one section describing the project classes and its methods, one paragraph as a user manual, and one paragraph describing the difficulties and more challenging parts.