COMP642 Advanced Programming
Semester 2 2024

Object Oriented Programming Software Design

Worth:	20%
Due:	**Friday, 27 September 2024 5:00 p.m.**
Late Penalty:	Work not received by the due time attracts an immediate penalty of up to 25% of the marks available. No work will be accepted after **Sunday, 29 September 2024 5:00 p.m**.
Submission:	Zip your completed files and submit the .zip through the link on COMP642 Akoraka | Learn page.

This is an **individual** assessment. You must not collaborate or confer with others. You may help others by verbally explaining concepts and making suggestions in general terms, but without directly showing or sharing your own code. You must develop the logical structure, the detail of your code and the database on your own, even if you are working alongside others. Code that is copied or shares a similar logic to others will receive zero marks for both parties.

The use of Artificial Intelligence (AI) tools, such as ChatGPT, to complete this assessment is **prohibited**. Assessment answers will be analysed for evidence of the use of AI and penalties may be administered.

The University policy on Academic Integrity can be found here.

Introduction

This software project is divided into two parts, Software Design and Software Development. For the first part of the project, your task is to design the Model classes. Your design must be developed according to good programming principles that will allow for reasonable subsequent changes to be made easily.

In the second part of this project, you will code your design (possibly amended after feedback) and you will create a GUI (using tkinter or Flask and Python) that works with that design. You will then write test cases to verify the functionalities of your software application. Part two of the project will be released separately.

Scenario

Fresh Harvest Veggies is a company based in Lincoln that sells high quality farm-fresh vegetables. The selection includes leafy greens, root vegetables, seasonal specials and herbs and aromatics. Since its number of customers is growing, it is keen to move from a manual system to an online order management system. In this initial stage, staff will process the order on behalf of the customers.

Customers can order vegetables by units of specified bunches, weights, punnet or pack. Customers can also purchase a premade box. The premade box is available in three sizes, small, medium and large. The content of the premade box can be customised and may vary based on the availability of vegetables. Fresh Harvest Veggies serves two types of customers: private customer and corporate customer. Private customers cannot place orders if the amount owing is greater than $100.00. Corporate customers cannot place orders if their balance is less than their credit limit. They also get a discount of 10% for each order.

When placing an order, customers can select individual items either by unit, weight, pack or premade box. For premade box, they can specify the size of the box and the number of boxes required. Customers may choose to either collect their order in person or opt for delivery. Delivery is available only within a 20-kilometer radius and is subject to a fixed fee of $10.00. During checkout, customers have the option of paying by credit card, debit card or charge to their account. If the amount is charged to their account, they should have the option for multiple payments to settle their account.

The completed system should allow the staff to place an order on behalf of the customer. When ordering, staff should be able to select individual items or the premade box, check out and process the payment. The system should accommodate payment from customers any time. Staff should be able to view customers' details, including the balance and order history. The system should also provide reporting for the company. This includes the total sales of the week, month and year, the list of all private and corporate customers and the most popular and unpopular items.

Requirements

Design the model classes based on the given scenario. DO NOT WRITE the code, just give the declarations of the classes. You will also need to provide a class diagram to show the attributes, methods and relationships between classes.

You need only as many methods as are required based on the scenario. You do not have to provide methods for all the possible reporting and updating that a complete system would need.

You must:

- Give the **signature of all methods** including constructors (provide the method name, parameter names, and the return value. Use type hints).
- List all the attributes for each class.
- Generate program documentation for your Python project using doxygen (see lecture video on Python documentation on our LEARN Course Page).

For example, in the Customer class you might give the signature for a method that returns the full name of the customer as follows:

```
def fullName(self) -> str:
    """! The method fullName returns the full name of the customer
    by concatenating the first and last name
    @param None
    @return a string that contains the full name of the customer
    """
    pass
```

The Python Dosctrings provide the description of the method indicating the parameters and the return value of the method. **You do not need to define the method**. This will be done in Part 2 of the project.

Do not hand in completed code. Just provide skeleton Python code (model classes and controller) commented using doxygen comments formatting style, the html folder that contains the generated HTML documentation and a class diagram. You should zip these files before uploading to Akoraka | LEARN.

Marking Criteria

Criteria	**Marks (out of 100)**	**Mark Range**
Class Diagram	**30**	Each class in the class diagram must show the class attributes and its relationship to other classes (composition, inheritance or collections). A good design (80% – 100%) is a design that fulfils all the user requirements and making good use of OOP concepts of encapsulation, abstraction, inheritance and polymorphism. An acceptable design (51% – 79%) fulfils most of the user requirements and makes good use of OOP concepts. A poor design (1% - 50%) only covers a few or some of the user requirements and demonstrates no or a poor use of OOP concepts on, inheritance or collections).
Model Classes	**70**	For each class, define the __init__() method and other appropriate methods. Each method in the class, must have method signature and appropriate comments to briefly describe the purpose of the method.
Total	**100**	