# COMP642 Advanced Programming
# Semester 2 2024

## Object Oriented Programming Assignment 2

| | |
|---|---|
| **Worth:** | 25% |
| **Due:** | **Friday, 13 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, 15 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.

## Scenario

Lincoln Office Supplies is a company based in Lincoln that supplies high quality office products and stationery. You have been asked to develop an application in Python using an object-oriented programming (OOP) approach to manage customers, orders, and payments for Lincoln Office Supplies. Once implemented, the application must be able to:

- Display a given customer's information.
- Create and process orders for existing customers.
- Accept payments from customers.
- List all orders for a given customer.
- List all payments from a given customer.
- List all customers.
- List all orders.
- List all payments.

Your design should be based on the Model View Controller (MVC) pattern:

- The model deals with the information about customers, orders, order items, products, and payments.
- The controller class should look after creating and maintaining objects in the model.
- The system must support the development of different views in which a user can interact with the system (such as forms, web forms, or data from files).

## The Model

The class diagram for this application is shown in Figure 1. A customer can place one or more orders, and each order can consist of one or more OrderItem objects. The OrderItem class specifies the product and the quantity being ordered, with the product details described in the Product class. For example, Customer 1001 can place an order for 2 boxes of blue ballpoint pens, 3 scissors, and a box of blue whiteboard markers.

A customer can also make one or more payments. For this assignment, you do not need to implement the delete functionality for any objects, and you can assume that product names are distinct. In addition, assume that orders and payments are for existing customers only.

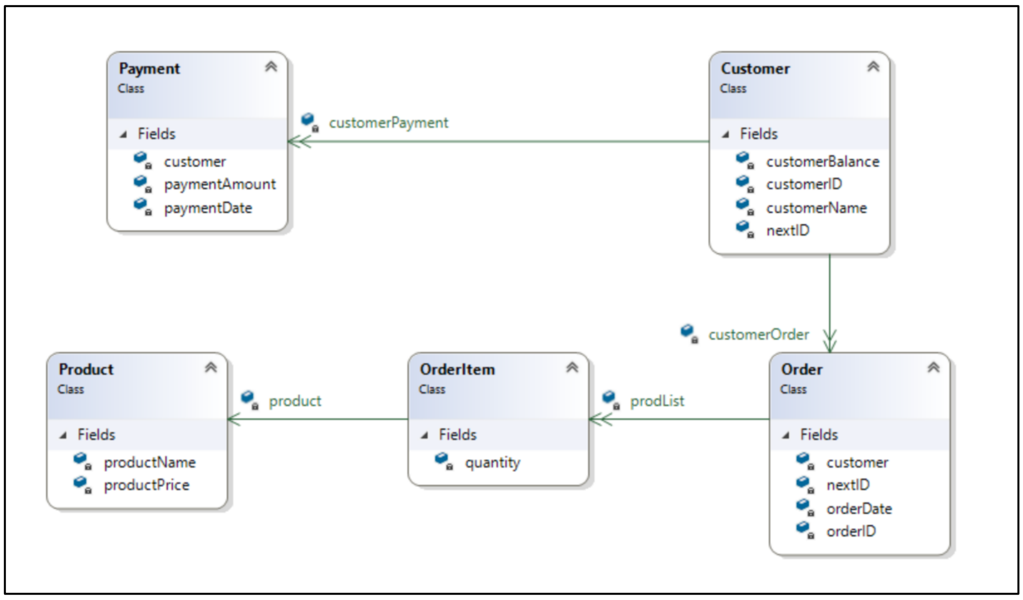


Figure 1- Class Diagram

## Visibility of Properties

- All the classes should have suitable constructors with parameters for those properties that are not set automatically.
- Unique IDs should be assigned automatically to any customer or order when a customer or an order object is created (start the first customer with ID 1000 and the first order with ID 10000).
- Payments and orders should be automatically assigned the current date when the objects are created.
- All the properties of each class should be available to a calling program.

## Company Controller Class

You will need a Company class. There will be only a single instance of this class and this will act as the controller. It should have public methods that will be available to different views and include methods to:

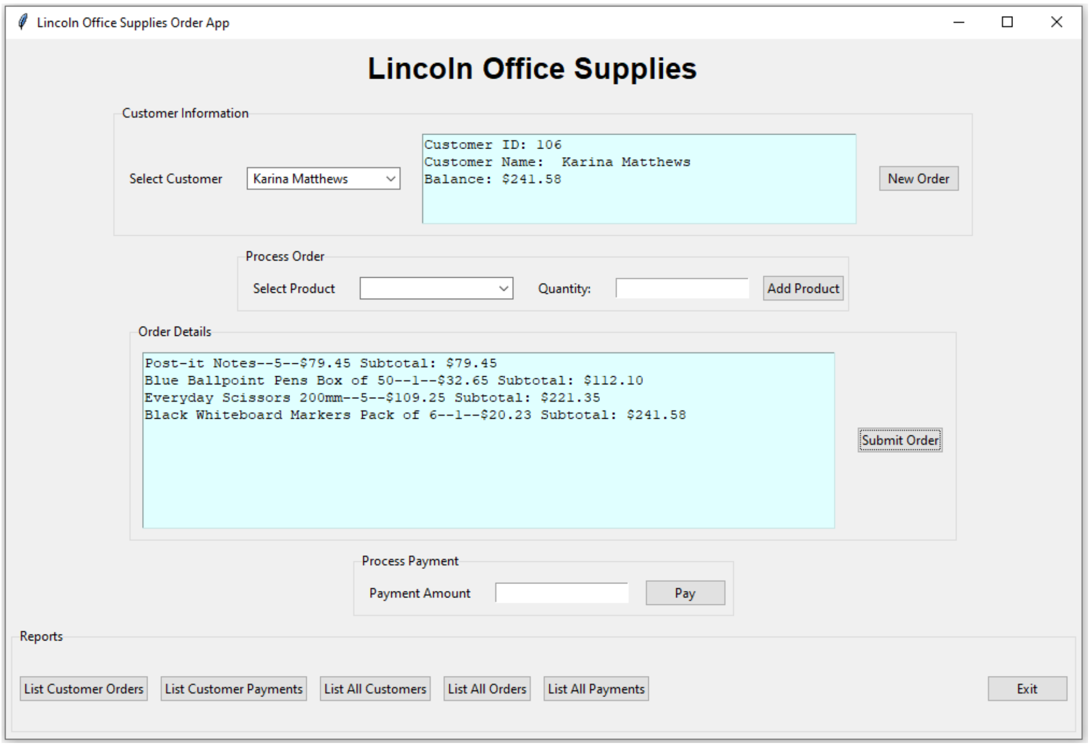
- Create and maintain customers, orders, products and payments.
- Find a customer object based on customer's name.
- Find a product object based on product's name.
- Add an order for a given customer.
- Add an order item for a given order.
- Add a payment for a given customer.
- Provide the list of orders for a given customer.
- Provide the list of payments for a given customer.
- Provide a list of all customers.
- Provide a list of all orders.
- Provide a list of all payments.

## Requirements

1. Your application must provide the following functionalities:
   a. Read the supplied files and create the appropriate customer and product objects. This information must be made visible to the user.
   b. Create a new order for a selected customer.
   c. Add products and quantities (as order items) to the order. Current total of the order must be visible to the user.
   d. Completed order must be submitted and the customer's balance must be updated.
   e. Make a payment for a selected customer. Customer's balance must be updated to reflect this payment.
   f. Display the list of orders for a selected customer.
   g. Display the list of payments for a selected customer.
   h. Display the list of all customers for the company.
   i. Display the list of all the orders for the company.
   j. Display the list of all payments for the company.
   k. Have a user interface with the appropriate user controls, useful feedback, and prevention of input errors.

2. Your code must:
   a. Make appropriate use of the model classes provided based on the class diagram shown in Figure 1.
   b. Be well-structured and easy to maintain.
   c. Be appropriately commented.
   d. Each class must provide getter and setter methods and any other appropriate methods.

3. Implement an appropriately designed view (GUI interface) using tkinter. An example is shown in Figure 2. **Note: You do not have to develop the exact replica of this interface. Feel free to design your own user interface**.



```
Customer ID: 106
Customer Name:  Karina Matthews
Balance: $241.58
```

```
Post-it Notes--5--$79.45 Subtotal: $79.45
Blue Ballpoint Pens Box of 50--1--$32.65 Subtotal: $112.10
Everyday Scissors 200mm--5--$109.25 Subtotal: $221.35
Black Whiteboard Markers Pack of 6--1--$20.23 Subtotal: $241.58
```

Figure 2- Possible Interface

## Marking Criteria

| **Criteria** | **Marks (out of 100)** | **Mark Range** |
|---|---|---|
| **Product** Class | **5** | Marks will be assigned using the following criteria:<br><br>All requirements met (81% - 100%)<br><br>Some requirements met (51% - 80%)<br><br>Minimum requirements met (1% - 50%) |
| **Payment** Class | **5** | |
| **OrderItem** Class | **5** | |
| **Order** Class | **10** | |
| **Customer** Class | **15** | |
| **Controller** Class | **25** | |
| View (User Interface) | **15** | |
| Error Handling | **10** | All relevant errors are detected and handled appropriately (9 – 10).<br><br>Some errors are detected and handled appropriately, but may miss some less common issues (5 – 8).<br><br>Minimal or ineffective error detection; many errors are not identified (0 – 4). |
| Coding style, comments, and clear logic | **10** | Excellent coding style, relevant comments, and clear logic (8 -10).<br><br>Acceptable coding style with some comments, and logic is reasonable (5 -7).<br><br>Poor coding style, minimal comments, and unclear logic (1- 4). |
| **Total** | **100** | |