UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 1

Object-oriented Programming

Submitted by:

Ruperto, April Anne A.

Instructor:

Engr. Maria Rizette H. Sayo

August 2, 2025

I. Objectives

This laboratory activity aims to implement the principles and techniques in object-oriented programming specifically through:

- Identifying object-orientation design goals
- Identifying the relevance of design patterns to software development

II. Methods

- Software Development
  - The design steps in object-oriented programming
  - Coding style and implementation using Python
  - Testing and Debugging
  - Reinforcement of the exercises below

A. Suppose you are on the design team for a new e-book reader. What are the primary classes and methods that the Python software for your reader will need? You should include an inheritance diagram for this code, but you do not need to write any actual code. Your software architecture should at least include ways for customers to buy new books, view their list of purchased books, and read their purchased books.

B. Write a Python class, Polygons that has three instance variables of type str, int, and float, that respectively represent the name of the polygon, its number of sides, and its area. Your class must include a constructor method that initializes each variable to an appropriate value, and your class should include methods for setting the value of each type and retrieving the value of each type.

III. Results

A.

PRIMARY CLASSES AND METHODS

1. User

- **Attributes:**
  - user_id
  - name
  - library (a list of purchased books)

- **Methods**
    - buy_book(book)
    - view_library()
    - read_book(book)
    -

2. Book

- **Attributes:**
    - Title
    - Author
    - Content
    - Price
- **Methods**
    - get_summary()
    - get_content()

3. Store

- **Attributes:**
    - available_books (a list of all books in the store)
- **Methods:**
    - search_book(title)
    - purchase_book(user, book)
    - list_books()

4. ReaderApps

- **Attributes**
    - current_user
    - store
- **Methods:**
    - start_session(user)
    - browse_store()
    - read(book)

5. EBook ***(inherits from Book)***

- **Additional Attributes:**
  - current_page
- **Additional Methods:**
  - turn_page()
  - go_to_page(page_number)

INHERITANCE DIAGRAM

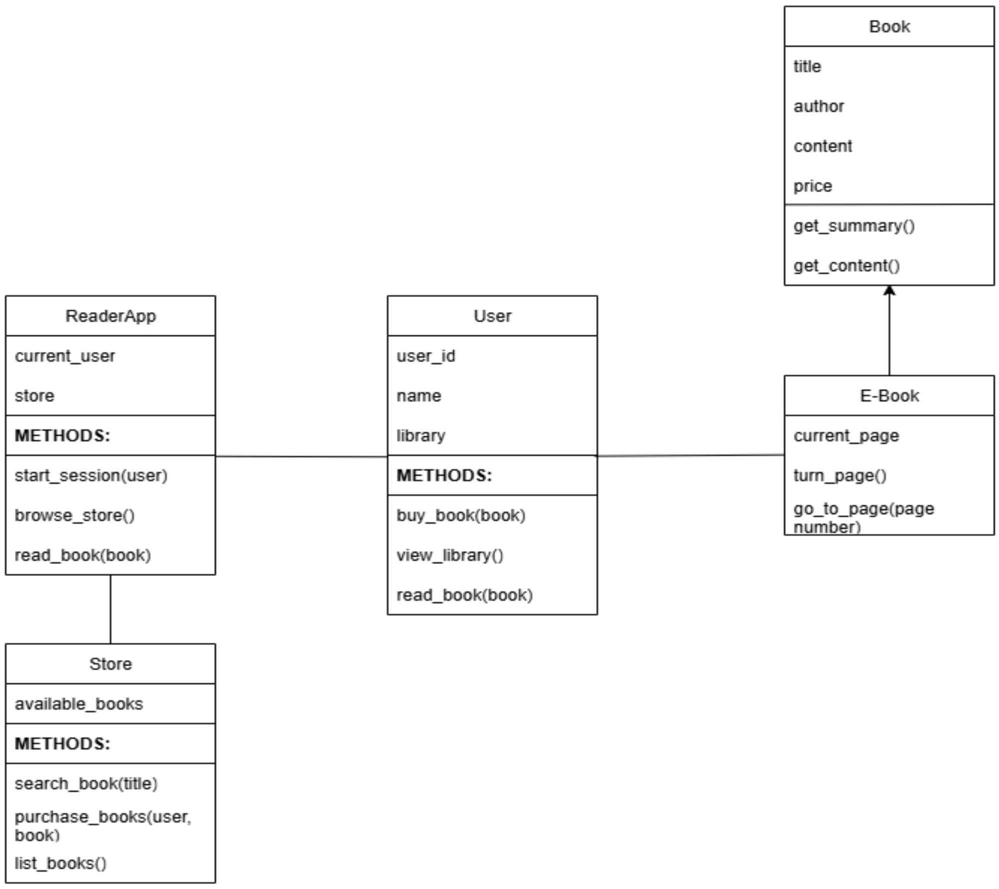

Figure 1: Screenshot of Diagram

This inheritance diagram shows how each class connects with each other. Each class is separated and labeled, with corresponding methods below. User interacts with both ReaderApp class and E-Book class. The ReaderApp class interacts with Store class where the user will purchase books. E-Book class inherits from Book class, shown with an arrow pointing to Book class.

B.

PRIMARY CLASS AND METHODS

Primary Class

- Polygon
    - Name
    - Sides
    - Area

Methods

- set_name(self, name: str)
- set_sides(self, sides: int)
- set_area(self, area: float)
- get_name(self) -> str
- get_sides(self) -> int
- get_area(self) -> float

INHERITANCE DIAGRAM

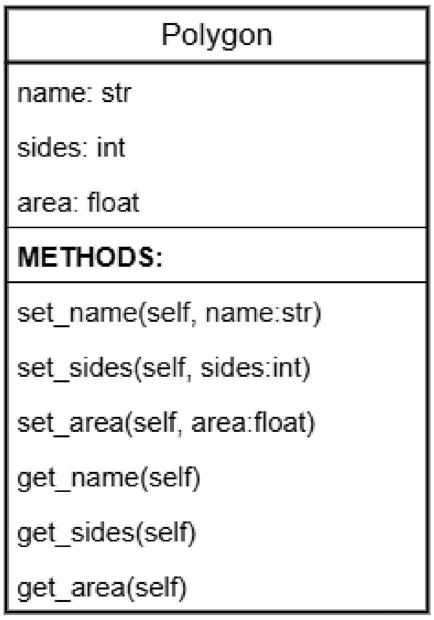

Figure 2: Screenshot of Diagram

In this inheritance diagram, classes are used for creating objects that share common properties and behaviors. Inside the main class, which is Polygon class, we will store pieces of information in each polygon which is name, sides and area. These methods encapsulate access to the internal variables and give you controlled access to the object's state. We used each method to update or change the object's attributes and to retrieve values.

3. Conclusion

This lab exercise emphasizes the need for a carefully planned class design, the concept of data encapsulation via attributes and methods, and the application of inheritance for code reuse and logical hierarchies (e.g., EBook inheriting from Book). The UML diagrams that went along not only helped visualize the relationships among different classes but also improved the understanding of the code's clarity and maintainability. Through the designing of constructors, setters, and getters, we discovered how classes co-operate in a program and how inheritance facilitates the function of related objects. In summary, this task connected our algorithmic thinking, software designing, and implementation capabilities strengthened our understanding on how to be utilized to represent real-world systems in a clean, efficient, and scalable fashion.

References

[1] Co Arthur O.. “University of Caloocan City Computer Engineering Department Honor Code,” UCC-CpE Departmental Policies, 2020.