UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 1

Object-oriented Programming

Submitted by:
Balaoro, Judge Wayne B.

Instructor:
Engr. Maria Rizette H. Sayo

July 26, 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 pattern 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 below exercises

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

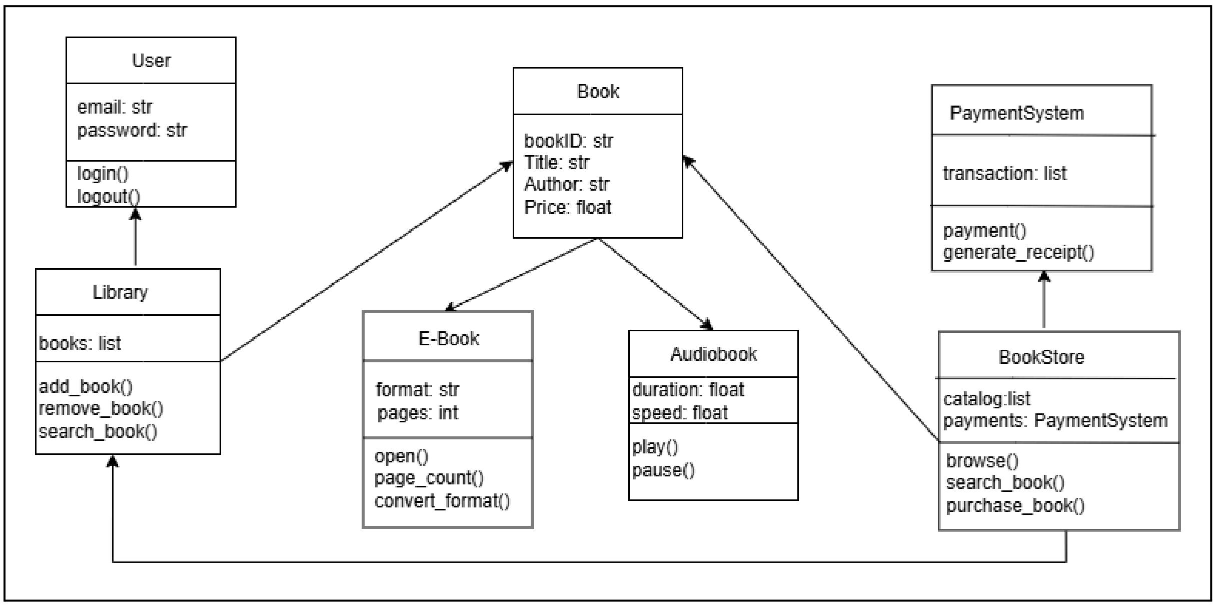
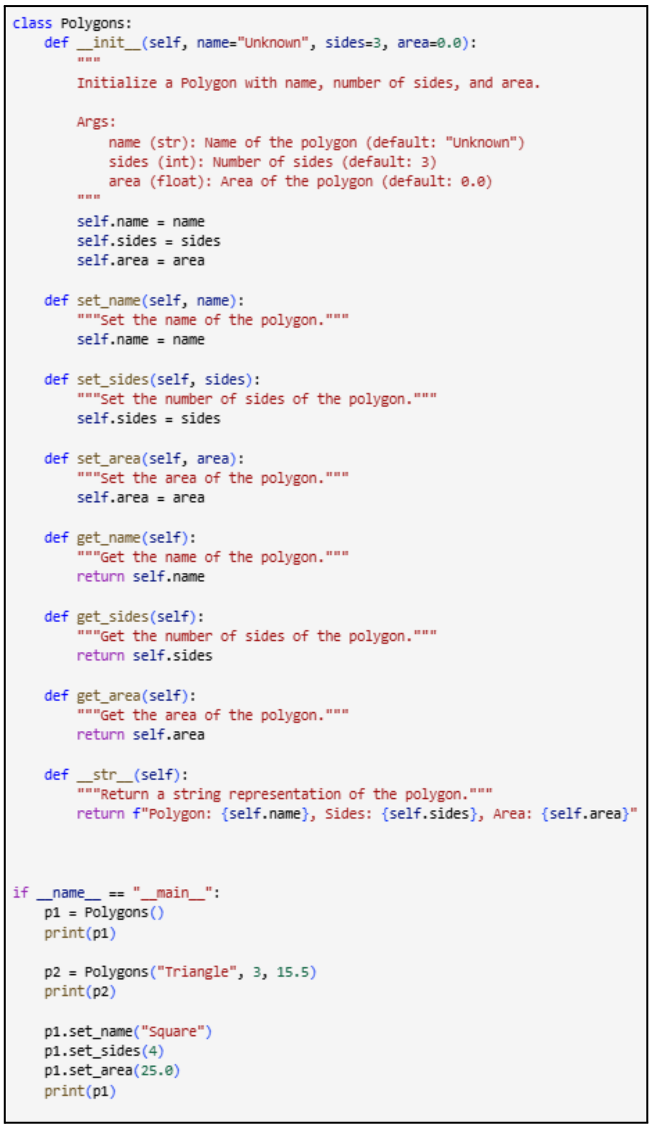


Figure 1. Inheritance Diagram

In this activity, I created a diagram that uses object-oriented programming to show the inheritance hierarchy in the e-book reader system. The User class acts as the foundation, managing tasks such as login and logout. This ensures secure access to users. The real heart of the system lies in the Book class. By making the Book the parent class with common attributes like title and author. I've made a flexible base. The two that inherits this class is E-book and Audiobook class and each adding their unique features whether it is handling different file formats for e-books or playback for audiobooks.

The Library class offers simple ways to add, remove and search for books. I kept this separate from the BookStore class which focuses on the business aspect of browsing and buying. This way the system is easier to maintain because if we need to change how purchases work later on, this won't affect the library functionality. In accordance with the idea that every class should have a single duty, the PaymentSystem manages transactions independently. Because of its modular design, we might be able to replace the payment processor without affecting other system components.



```python
class Polygons:
    def __init__(self, name="Unknown", sides=3, area=0.0):
        """
        Initialize a Polygon with name, number of sides, and area.

        Args:
            name (str): Name of the polygon (default: "Unknown")
            sides (int): Number of sides (default: 3)
            area (float): Area of the polygon (default: 0.0)
        """
        self.name = name
        self.sides = sides
        self.area = area

    def set_name(self, name):
        """Set the name of the polygon."""
        self.name = name

    def set_sides(self, sides):
        """Set the number of sides of the polygon."""
        self.sides = sides

    def set_area(self, area):
        """Set the area of the polygon."""
        self.area = area

    def get_name(self):
        """Get the name of the polygon."""
        return self.name

    def get_sides(self):
        """Get the number of sides of the polygon."""
        return self.sides

    def get_area(self):
        """Get the area of the polygon."""
        return self.area

    def __str__(self):
        """Return a string representation of the polygon."""
        return f"Polygon: {self.name}, Sides: {self.sides}, Area: {self.area}"


if __name__ == "__main__":
    p1 = Polygons()
    print(p1)

    p2 = Polygons("Triangle", 3, 15.5)
    print(p2)

    p1.set_name("Square")
    p1.set_sides(4)
    p1.set_area(25.0)
    print(p1)
```

Figure 2. Source Code

In this second activity, I created a source code managed by the Polygons class. Each polygon's name is stored as text, the number of sides is stored as a whole number, and the area is stored as a decimal number. When you create a new polygon, it will automatically sets the default values. (a 3 sided "Unknown" polygon with 0.0 area) if you don't specify them, which makes it more convenient to use. By keeping the data accessible and organized, the class is following good programming practices. When printing a polygon object, a special string method neatly formats the data.

IV. Conclusion

In this lab activity, I was able to apply an important object-oriented programming concept that I learned last semester in this lab activity. Using inheritance, I first created an e-book reader system to create a logical structure for user, book management and payment processing. Also, by using getter and setter methods, I developed a Polygons class that demonstrates a proper use of encapsulation/

Through these two exercises, I gained hands-on experience with core OOP principles including inheritance, encapsulation, and modular design. All that has been tackled on the last semester course of Object-Oriented Programming. The architecture of the e-book reader demonstrates how to organize complex systems, while the Polygons class illustrates fundamental class design techniques. Both implementations use Python's object- oriented features while keeping their code structures clear and manageable.

References

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

[2] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design patterns: Elements of reusable object-oriented software. Addison-Wesley.

[3] Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media.