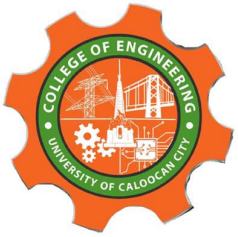
UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT



Data Structure and Algorithm

Laboratory Activity No. 1

Object-oriented Programming

Submitted by:
Asugas, Kenneth R.

Instructor:
Engr. Maria Rizette H. Sayo

07, 28, 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

A,

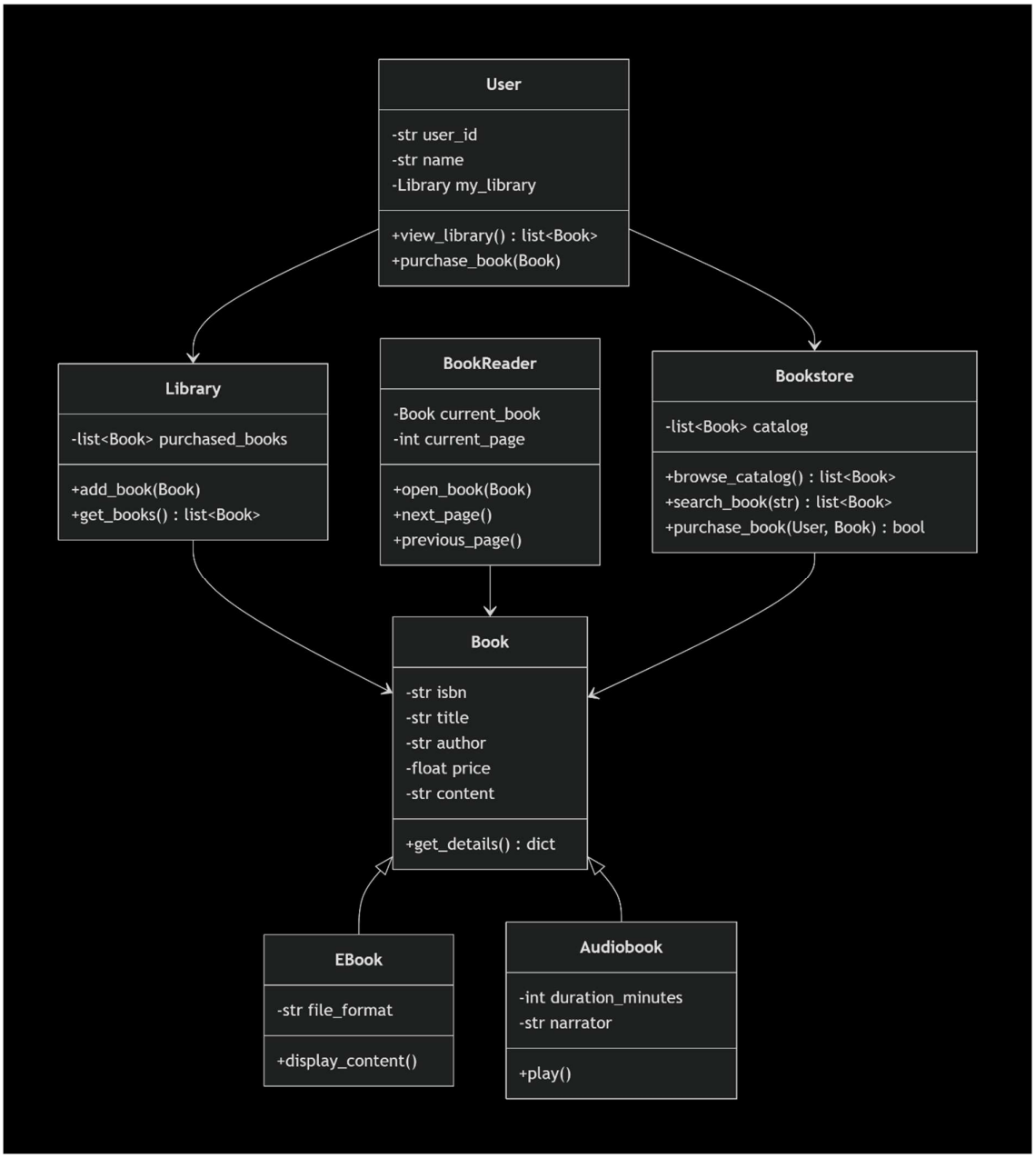


Figure 1 Screenshot of diagram

B.

```python
class Polygons:
    """A class to represent a geometric polygon."""

    def __init__(self, name: str = "Unknown", sides: int = 0, area: float = 0.0):
        """
        Constructor to initialize the Polygons object.
        :param name: Name of the polygon (str)
        :param sides: Number of sides (int)
        :param area: Area of the polygon (float)
        """
        self._name = name
        self._sides = sides
        self._area = area

    def get_name(self) -> str:
        """Retrieves the name of the polygon."""
        return self._name

    def get_sides(self) -> int:
        """Retrieves the number of sides."""
        return self._sides

    def get_area(self) -> float:
        """Retrieves the area."""
        return self._area

    def set_name(self, name: str):
        """Sets the name of the polygon."""
        self._name = name

    def set_sides(self, sides: int):
        """Sets the number of sides."""
        self._sides = sides

    def set_area(self, area: float):
        """Sets the area of the polygon."""
        self._area = area

if __name__ == "__main__":
    triangle = Polygons("Triangle", 3, 15.5)

    print(f"Initial Polygon: {triangle.get_name()}, Sides: {triangle.get_sides()}, Area: {triangle.get_area()} sq units")

    triangle.set_name("Equilateral Triangle")
    triangle.set_area(20.1)

    print(f"Updated Polygon: {triangle.get_name()}, Sides: {triangle.get_sides()}, Area: {triangle.get_area()} sq units")
```

Figure 2 Screenshot of program

```
Initial Polygon: Triangle, Sides: 3, Area: 15.5 sq units
Updated Polygon: Equilateral Triangle, Sides: 3, Area: 20.1 sq units
```

Figure 3 Output of program

IV. Conclusion

This lab report successfully reinforced the principles and techniques of object-oriented programming. The first exercise involved designing a software architecture for an e-book reader, which provided practical experience in identifying primary classes, their responsibilities, and their interrelationships using an inheritance diagram. This process highlighted the importance of OOP design goals such as reusability, scalability, and maintainability. The second exercise involved the hands-on implementation of a Polygons class in Python, which solidified understanding of core OOP concepts like encapsulation through the use of constructors, instance variables, and getter/setter methods. Together, these exercises demonstrated how a well-structured, object-oriented approach is fundamental to developing robust and organized software systems.

References

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