



UNIVERSITY OF CALOOCAN CITY  
COMPUTER ENGINEERING DEPARTMENT



Data Structure and Algorithm

Laboratory Activity No. 1

---

# Object-oriented Programming

---

*Submitted by:*  
Disomnong, Jalilah M.

*Instructor:*  
Engr. Maria Rizette H. Sayo

07-26-2025

## I. Objectives

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

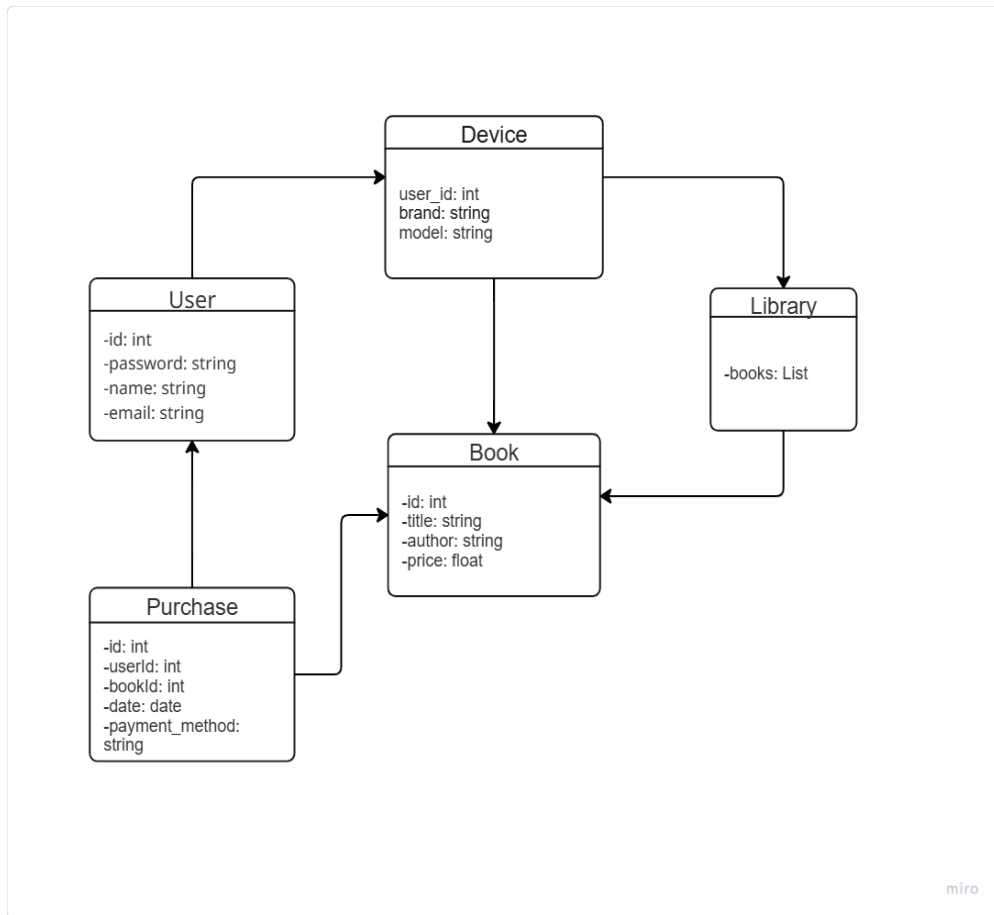
- Identifying object-orientation design goals
- Identifying the relevance of design pattern to software development

## 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

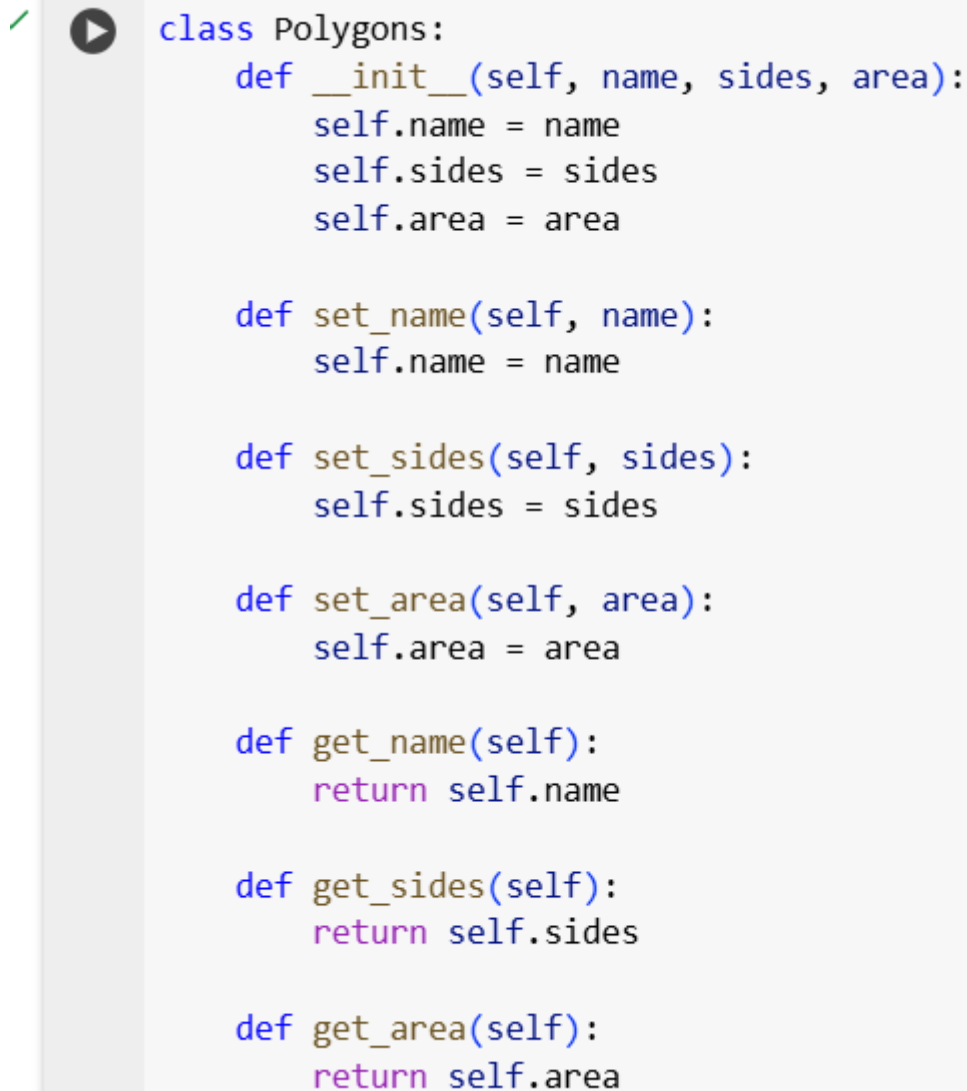
- A. In this section, the diagram below shows the structures of the design for the e-book reader.



*Figure 1 Diagram for e-book reader*

The class diagram outlines how different parts of the eBook reader system work together. The User class represents individuals who can purchase and access books. The Book class holds information about each book, while the Purchase class connects users and books, recording transactions. The Library stores the books that users have access to, and the Shop (or eBook platform) serves as the main environment where users interact with the system. Each class is designed to focus on a specific role, making the system organized and easy to manage. The relationships between classes clearly show how data flows from purchasing to storing and reading books.

B. In this section, it shows that the Python Class was created to represent the properties of a polygon. It includes the attributes for the polygon's name, number of sides, and area. The figures below show how the class is used to set and retrieve these values.



```
class Polygons:
    def __init__(self, name, sides, area):
        self.name = name
        self.sides = sides
        self.area = area

    def set_name(self, name):
        self.name = name

    def set_sides(self, sides):
        self.sides = sides

    def set_area(self, area):
        self.area = area

    def get_name(self):
        return self.name

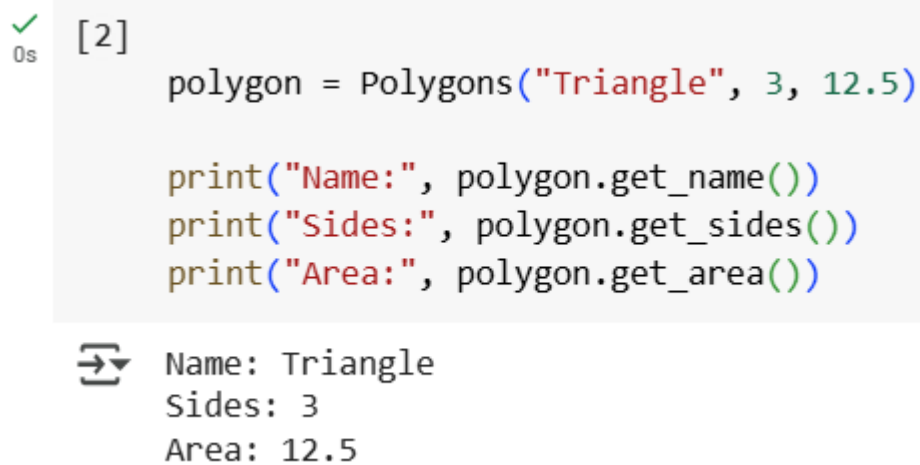
    def get_sides(self):
        return self.sides

    def get_area(self):
        return self.area
```

Figure 2 Polygon Class refer to this link: <https://colab.research.google.com/drive/1rM-RYnOULpd05RIT32t76kGMVD6it-pe>

In figure 2, the code defines a class called Polygons, which is used to represent a polygon with three main properties: its name, the number of sides, and its area. When a new object from this class is created, the `__init__` method is automatically called to initialize these three values. The class includes setter methods—`set_name`, `set_sides`, and `set_area`—which allow the user to change or update the values of the name, number of sides, and area after the object has been created. It also includes getter methods—`get_name`, `get_sides`, and `get_area`—which return the current values stored in the object. This design makes it easy to store, modify, and retrieve information about a polygon in an organized way. The code demonstrates basic

principles of object-oriented programming, such as encapsulation and the use of methods to manage object data (GeeksforGeeks, 2023; Real Python, 2022).



```
[2]
polygon = Polygons("Triangle", 3, 12.5)

print("Name:", polygon.get_name())
print("Sides:", polygon.get_sides())
print("Area:", polygon.get_area())
```

⇒ Name: Triangle  
Sides: 3  
Area: 12.5

Figure 3 Example refer to this link: <https://colab.research.google.com/drive/1rM-RYnOULpd05RIT32t76kGMVD6it-pe>

In figure 3, the object polygon is created from the Polygons class with the values “Triangle” as the name, 3 as the number of sides, and 12.5 as the area. These values are passed to the constructor method `__in__` and stored in the object. The output indicated that the object was successfully created and the values were correctly stored and retrieved. It shows that the class is working as expected by allowing data to be accessed using a method.

## IV. Conclusion

In conclusion, this laboratory activity is about learning and applying the basic ideas of object-oriented programming (OOP). It focuses on how to design and build programs using objects, which are the building blocks of OOP. In this activity, we created a design for an e-book reader using class diagrams, and we also wrote a Python class to represent a polygon. These tasks helped us understand how to organize a program using classes, how to create and use objects, and how OOP can make programs easier to manage and update.

## References

- [1] Google. (2024). Google Colaboratory. Retrieved April 18, 2024, from <https://colab.research.google.com/>
- [2] GeeksforGeeks. (2023). *Getter and Setter in Python*. Retrieved from <https://www.geeksforgeeks.org/getter-and-setter-in-python>
- [3] Real Python. (2022). *Python's property(): Add Managed Attributes to Your Classes*. Retrieved from <https://realpython.com/python-getter-setter>