UNIVERSITY OF CALOOCAN CITY
COMPUTER ENGINEERING DEPARTMENT

Data Structure and Algorithm

Laboratory Activity No. 1

Object-oriented Programming

Submitted by:
Talagtag, Mark Angel Tigue

Instructor:
Engr. Maria Rizette H. Sayo

July 26 2025

I. Objectives

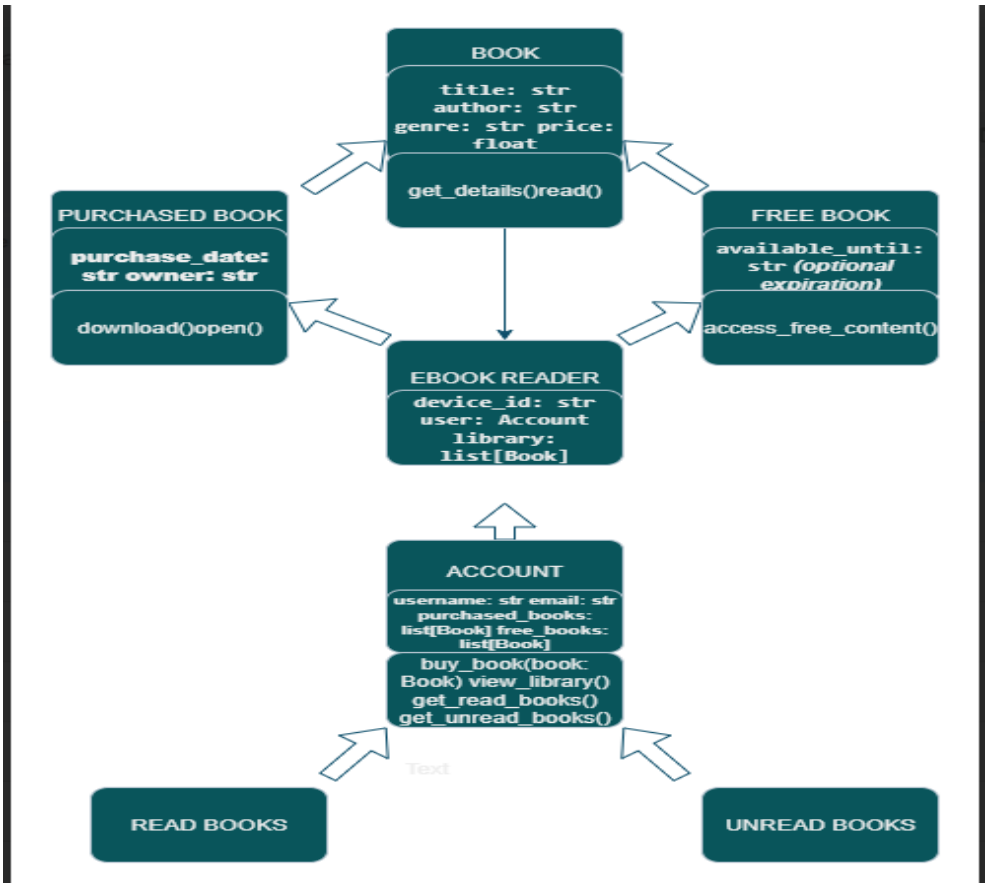
This laboratory activity aims to implement the principles and techniques in object-oriented programming specifically through:

- 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

Present the visualized procedures done. Also present the results with corresponding data visualizations such as graphs, charts, tables, or image . Please provide insights, commentaries, or explanations regarding the data. If an explanation requires the support of literature such as academic journals, books, magazines, reports, or web articles please cite and reference them using the IEEE format.

Please take note of the styles on the style ribbon as these would serve as the style format of this laboratory report. The body style is Times New Roman size 12, line spacing: 1.5. Body text should be in Justified alignment, while captions should be center-aligned. Images should be readable and include captions. Please refer to the sample below:

```
C:\Users\Administrator\PyCharmMiscProject\.venv\Scripts\python.exe "C:\Users\Administrator\PyCharmMiscProject\laboratory 1 dsa.py"
Name: Triangle
Sides: 3
Area: 15.5

Updated Polygon:
Name: Square
Sides: 4
Area: 25.0

Process finished with exit code 0
```

The Polygons class in Python is a simple yet meaningful representation of how object-oriented programming (OOP) principles can be used to simulate real-world objects—in this case, geometric shapes. It is designed to store and manipulate essential properties of a polygon: its name (as a string), number of sides (as an integer), and area (as a float). These three properties serve as the foundation for understanding and working with basic geometric figures in programming.

One key insight from the class design is the emphasis on **data encapsulation**. By using setter and getter methods, the class prevents direct access to its attributes, allowing more control over how values are modified. This practice not only improves data security but also makes future maintenance easier—validations or constraints can be added later without changing how the class is used.

Figure 1 Screenshot of program

If an image is taken from another literature or intellectual property, please cite them accordingly in the caption. Always keep in mind the Honor Code [1] of our course to prevent failure due to academic dishonesty.

IV. Conclusion

In software development, especially when using object-oriented programming, proper planning, structured design, and thoughtful implementation are essential to creating efficient and maintainable systems. By examining the steps of OOP design, applying consistent coding style in Python, and reinforcing the importance of testing and debugging, we can ensure the development process is both effective and adaptable.

In Exercise A, we explored the architecture for an e-book reader system by identifying essential classes and their relationships through an inheritance diagram. This demonstrates the value of planning before coding—understanding what components are needed, how they interact, and how users will navigate the software (buying, viewing, and reading books).

In Exercise B, we implemented a Polygons class in Python to show how OOP concepts like encapsulation, constructors, and access methods can be applied in practice. This reinforces the idea that even simple real-world objects can be modeled effectively through classes and instance variables.

Altogether, these exercises highlight the core practices of good software development: thoughtful design, clean code, and practical use of object-oriented principles. These are essential skills for building scalable, user-friendly applications in Python and beyond.

References

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