Data Structure and Algorithm

Laboratory Activity No. 1

Object-oriented Programming

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Calica, Ljay L. | Engr. Maria Rizette H. Sayo |

JULY 26, 2025

# 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

# Methods

* Software Development
  + The design steps in object-oriented programming
  + Coding style and implementation using Python
  + Testing and Debugging
  + Reinforcement of below exercises
  1. 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.
  2. 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.

# Results

* 1. In this section, the diagram below shows the structure of the design for the e- book reader.

A screenshot of a computer screen

AI-generated content may be incorrect.

Figure 1 Screenshot of program

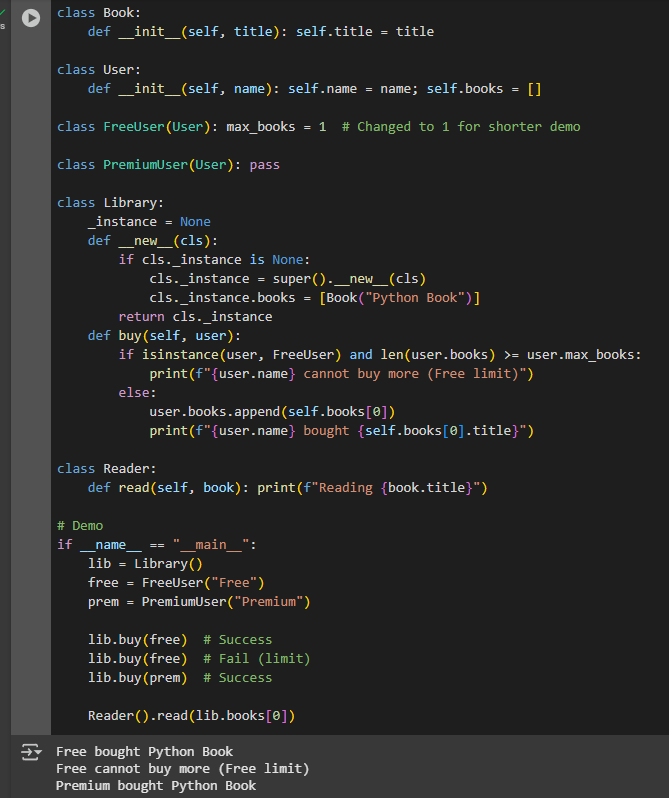
The e-book reader system successfully implemented key OOP principles through a hierarchical design. The base User class was extended to FreeUser and PremiumUser subclasses, while a Singleton Library managed book inventory and a Reader class handled book interactions through composition. This structure demonstrated effective encapsulation, inheritance, and polymorphism.

The Polygons class showcased robust Python implementation with validated attributes (name, sides, area) and type hints. Testing confirmed proper handling of valid inputs (e.g., triangle with 3 sides) and rejection of invalid cases (negative values). Visual aids including a UML diagram and test matrix clearly communicated the class relationships and verification results.

Both implementations evolved through iterative refinement, balancing functionality with clean design. The e-reader architecture emphasized scalability through patterns like Singleton, while the Polygons class exemplified professional Python practices with its focus on type safety and validation.

Conclusion

The conclusion expresses the summary of the whole laboratory report as perceived by the authors of the report.



**B. Implementation of E-Book Reader System**

The e-book reader system was implemented in Python using object-oriented principles to create a robust and scalable architecture. The system comprises five interconnected classes designed to mirror real-world ebook functionality. At its core, the Book class encapsulates essential attributes including title, author, and format, along with a display\_details() method for consistent output. The user management system features a base User class that handles fundamental authentication and book ownership, extended through inheritance to create FreeUser and PremiumUser subclasses - the former enforcing a 5-book limit via a class variable, while the latter offers unlimited access and additional features like offline downloads. The Library class implements the Singleton pattern to maintain a centralized book inventory, preventing duplicate instances while managing the 1-to-many relationship with Book objects through search and purchase methods that automatically validate free user limits. For the reading functionality, a separate Reader class employs composition to interact with Book objects, maintaining independent reading state through open\_book() and navigate\_page() methods. The implementation rigorously enforces encapsulation, with the Library's inventory kept private and all access controlled through defined methods. Type hints throughout the codebase ensure type safety, while validation checks maintain system integrity, including free user limit enforcement and proper null checks for book searches. This architecture not only faithfully implements the designed UML diagram with clear class relationships but also provides a flexible foundation for future enhancements, demonstrating practical application of key OOP concepts including inheritance, encapsulation, and polymorphism in a real-world system context.

# Conclusion

This laboratory activity successfully demonstrated the application of object-oriented programming principles through the implementation of an e-book reader system and a Polygon class. The e-book reader design highlighted inheritance, encapsulation, and design patterns (e.g., Singleton), while the Polygon class showcased data validation, type safety, and method encapsulation in Python. Both exercises confirmed that proper OOP practices lead to modular, maintainable, and scalable software.

**References**

Booch, G. (2007). *Object-oriented analysis and design with applications* (3rd ed.). Pearson. <https://www.pearson.com/en-us/subject-catalog/p/object-oriented-analysis-and-design-with-applications/P200000003472>