Data Structure and Algorithm

Laboratory Activity No. 1

Object-oriented Programming

|  |  |
| --- | --- |
| *Submitted by:* | *Instructor:* |
| Adoracion, Jerick Dave D. | 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

The following steps were implemented:

* 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

**A. E-Book Reader Software Design**

The primary classes and methods required for an e-book reader were enlisted, along with an image containing inheritance diagram.

A diagram of a book

AI-generated content may be incorrect.

*Figure 1 Diagram of eBook*

In this image, inheritance diagram represents the object-oriented design of an eBook reader system. At the top of the hierarchy is the parent class Books, which contains common properties and behaviors shared by all types of books, such as title, author, and publishing date. From this parent class, two subclasses inherit: Purchased Book and Free Book. These subclasses represent specific types of books and can have additional attributes unique to them—such as price and purchase date for purchased books, or promotional status for free books. The Ebook Reader class aggregates both types of books, serving as the central system where books are stored, accessed, and managed. Below the eBook reader is the Account class, which manages user-specific information and interactions with the reader. Each account keeps track of the user's reading activity through two categories: Read Books and Unread Books, indicating the reading status of the user's library. This structure showcases both inheritance and composition, making the system modular, organized, and scalable.

B. 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 areas. The figures below show how the class is used to set and retrieve these values

A screenshot of a computer program

AI-generated content may be incorrect.

*Figure 2: Source Code*

A white background with black and white clouds

AI-generated content may be incorrect.

*Figure 3: Output*

In figure 3, the object polygon is created from the Polygons class with the values “Hexagon” as the name, 6 as the number of sides, and 93.5 as the area.

# Conclusion

In this laboratory activity, we engaged in the structured design and conceptualization of functional software systems, focusing specifically on object-oriented programming (OOP) principles. The task of designing an e-book reader application allowed us to explore how to effectively structure a software system by decomposing it into modular, interacting classes. This exercise enhanced our understanding of class responsibilities, object relationships, and the importance of encapsulation and abstraction in system design.

Furthermore, implementing the Polygons class provided hands-on experience with defining custom objects, encapsulating properties, and implementing getter and setter methods. Through this, we gained practical insight into how classes can represent real-world entities and how their attributes and behaviors are managed programmatically.

**References**

IEEE. (2023). *IEEE citation guidelines*. [https://www.ieee.org](https://www.ieee.org/)

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

Lutz, M. (2013). *Learning Python* (5th ed.). O’Reilly Media.

McKinney, W. (2017). *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython* (2nd ed.). O’Reilly Media.

Python Software Foundation. (2023). *Python documentation*. <https://docs.python.org/3/>