Adamson University
College of Engineering
Computer Engineering Department

Data Structure and Algorithm

Laboratory Activity No. 1

Object-oriented Programming

Submitted by:
Manigbas, Jeus Miguel T. <Leader>
Barredo, Alwin P.
Bela, Lorenzo Miguel D.
Callorina, Robert Victor A.
Guzon, Kean Louiz I.

Instructor:
Engr. Maria Rizette H. Sayo

November 08, 2023

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 have 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.

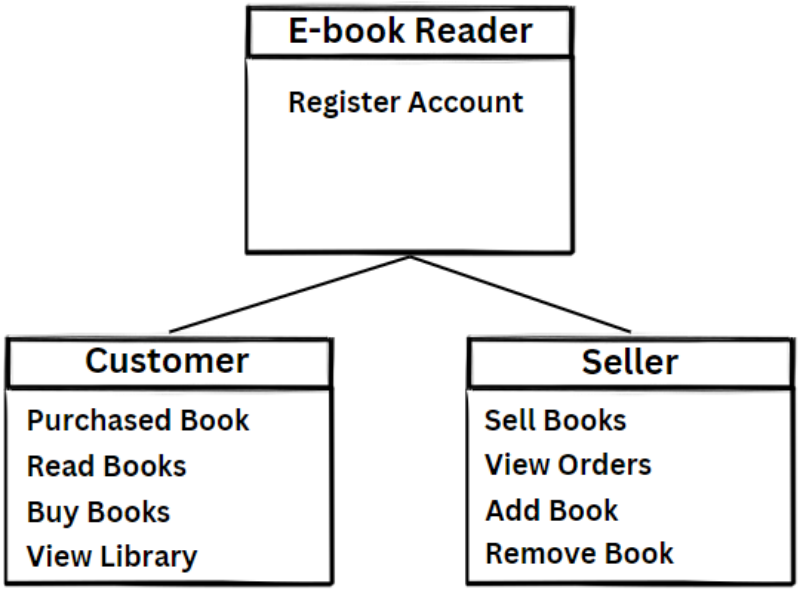

Figure 1. Screenshot of the inheritance diagram

This shows the developmental path of designing an e-Book reader using object-oriented programming principles. Inheritance is a key concept in object-oriented programming that models the relationship between two classes and enables code reuse.

B.

```python
import math

class Polygon:

  def __init__(self, name, num_sides, area):
    self.name = name
    self.num_sides = num_sides
    self.area = area

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

  def get_name(self):
    return self.name

  def set_num_sides(self, num_sides):
    self.num_sides = num_sides

  def get_num_sides(self):
    return self.num_sides

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

  def get_area(self):
    return self.area

  def initialize(self, num_sides):
```

Figure 2.1 Code

```python
    self.polygon_db = {
      3: "Triangle",
      4: "Square",
      5: "Pentagon",
      6: "Hexagon",
      7: "Heptagon",
      8: "Octagon",
      9: "Nonagon",
      10: "Decagon",
      11: "Hendecagon",
      12: "Dodecagon"
    }

    if num_sides in self.polygon_db:
      name = self.polygon_db[num_sides]
      self.set_name(name)

    else:
      print("Invalid number of sides")

  def get_info(self):
    base = float(input("Enter base: "))
    height = float(input("Enter height: "))
    sides = int(input("Enter number of sides: "))
    self.initialize(sides)
    area = base * height / 2
    self.set_area(area)
    print(self.get_name(), "with", sides, "sides has area", self.get_area())

polygon = Polygon("", 0, 0)
polygon.get_info()
```

Figure 2.2 Code

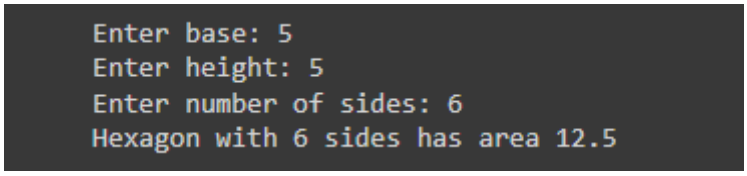

Figure 3. Output of the Code

The `Polygon` class represents a polygon shape with a name, number of sides, and area. It has methods to set and get the name, number of sides, and area. [2]

The `initialize` method initializes the polygon with the given number of sides. It uses a dictionary `polygon_db` to map the number of sides to the corresponding name of the polygon. If the number of sides is valid, it sets the name of the polygon using the `set_name` method. Otherwise, it prints an error message. [3]

The `get_info` method prompts the user to enter the base, height, and number of sides of the polygon. It then calls the `initialize` method to set the name of the polygon based on the number of sides. After that, it calculates the area of the polygon using the formula `base * height / 2` and sets the area using the `set_area` method. Finally, it prints the name, number of sides, and area of the polygon using the `get_name`, `get_num_sides`, and `get_area` methods.
The last two lines of the code create an instance of the `Polygon` class with an empty name, 0 number of sides, and 0 area. It then calls the `get_info` method to get the information of a polygon from the user. [4][5][6]

Overall, this code demonstrates the basic concepts of object-oriented programming in Python, such as classes, methods, attributes, and object instantiation. It also shows how to use dictionaries for mapping values and how to interact with the user through input and output.

IV. Conclusion

In conclusion, the given code demonstrates the fundamental concepts of object-oriented programming in Python, including classes, methods, attributes, and object instantiation. It also showcases the use of dictionaries for mapping values and how to interact with the user through input and output. The activity challenges us to think ahead of the project and highlights the importance of organizing the structure of the project before implementation. With a well-organized outline, software development becomes less complicated. One additional point to

note is that the Polygon class in the given code can be easily extended to include more functionality and attributes. For example, we could add methods to calculate the perimeter, angles, or other properties of the polygon. We could also add attributes to store the coordinates of the vertices, the color or style of the polygon, or other metadata. By using object-oriented programming, we can create modular and reusable code that can be adapted to different use cases and requirements. This makes the code more flexible, maintainable, and scalable, which are important qualities for software development.

References

[1] Co Arthur O.. “Adamson University Computer Engineering Department Honor Code,” AdU-CpE Departmental Policies, 2020.

[2] "Polygon," Math is Fun, [Online]. Available: https://www.mathsisfun.com/geometry/polygons.html. [Accessed: Nov. 8, 2023].

[3] "Python Classes and Objects," W3Schools, [Online]. Available: https://www.w3schools.com/python/python_classes.asp. [Accessed: Nov. 8 , 2023].

[4] "Modular programming," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Modular_programming. [Accessed: Nov. 8, 2023].

[5] "Inheritance and Composition: A Python OOP Guide," Real Python, 2021. [Online]. Available: https://realpython.com/inheritance-composition-python/. [Accessed: Nov. 8, 2023].

[6] "Python Dictionaries," W3Schools, [Online]. Available: https://www.w3schools.com/python/python_dictionaries.asp. [Accessed: Nov. 8, 2023].