UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

Object-Oriented Programming

Laboratory Activity No. 1

Review of Technologies

Submitted by:
Palmes, Lewis Clark L.
Saturday / BS CpE 1-A

Submitted to
Engr. Maria Rizzete Sayo
Instructor

Date Performed:
18-01-2025

Date Submitted
18-01-2025

I. *Objectives*

In this section, the goals in this laboratory are:

- *To define the key terms in Object-oriented programming*
- *To be able to know the construction of OO concepts in relation to other types of programming such as procedural or functional programming*

II. *Methods*

General Instruction: Define and discuss the following Object-oriented programming concepts:

1. ***Classes*** - *A class is a template or blueprint that defines the structure and behavior (in terms of fields and methods) that the objects created from the class will have. It is a way of bundling data and methods that operate on that data into one single unit. Classes support encapsulation, which helps in hiding the internal implementation and exposing only the functionalities. For example, a class 'Car' might contain fields like 'polo', 'mini', and 'beetle', and methods like 'start()' and 'stop()'.*

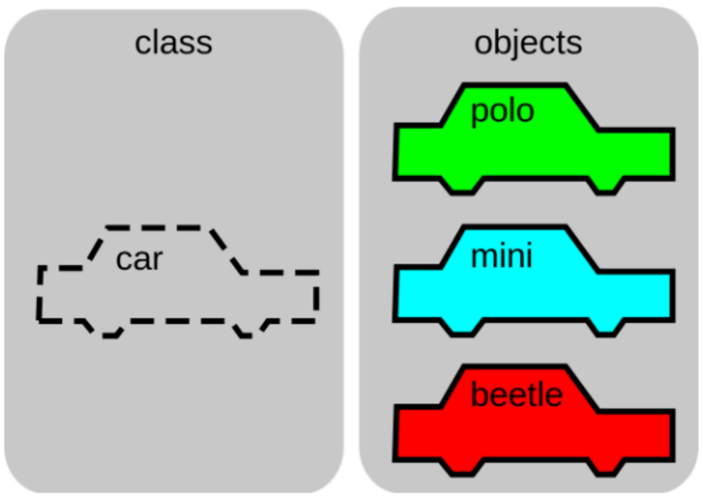
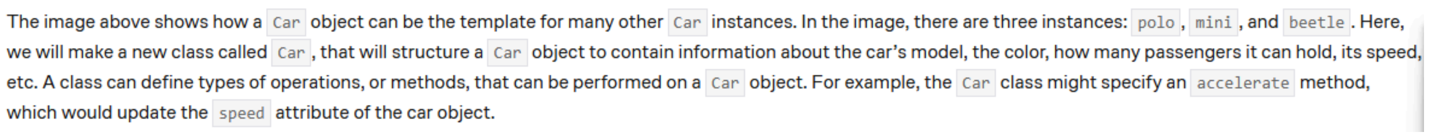


Figure 1: [1] ***Class Figure***



The image above shows how a `Car` object can be the template for many other `Car` instances. In the image, there are three instances: `polo`, `mini`, and `beetle`. Here, we will make a new class called `Car`, that will structure a `Car` object to contain information about the car's model, the color, how many passengers it can hold, its speed, etc. A class can define types of operations, or methods, that can be performed on a `Car` object. For example, the `Car` class might specify an `accelerate` method, which would update the `speed` attribute of the car object.

2. ***Objects*** **-** *An object is an instance of a class, and it represents a specific entity that contains the data defined by the class's fields and the behaviors defined by the class's methods. Every object created from a class has its unique set of data, although the structure and behavior are defined by the class.*

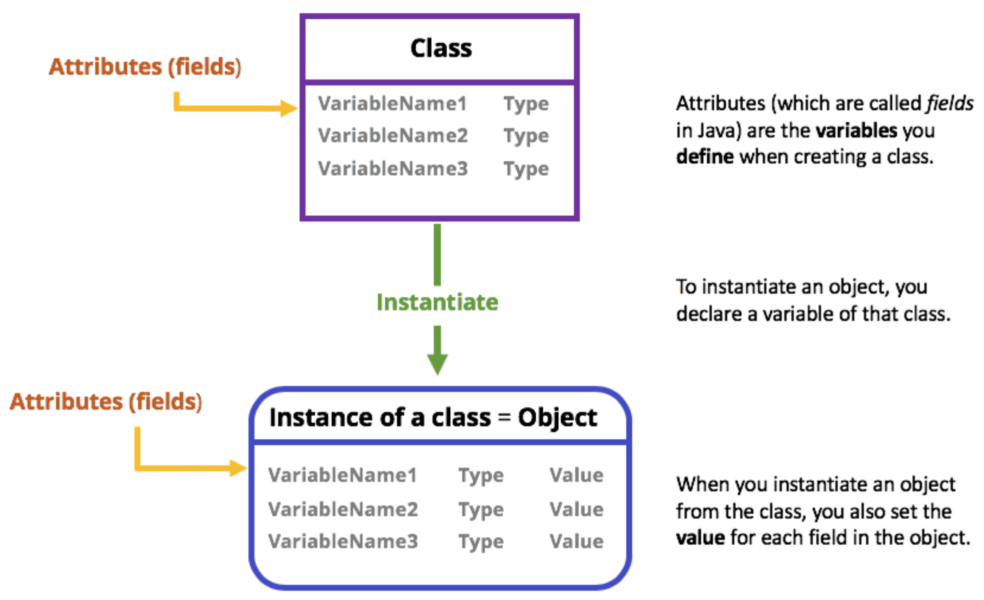

Class
VariableName1 Type
VariableName2 Type
VariableName3 Type
Attributes (fields)
Attributes (which are called fields in Java) are the variables you define when creating a class.
Instantiate
To instantiate an object, you declare a variable of that class.
Attributes (fields)
Instance of a class = Object
VariableName1 Type Value
VariableName2 Type Value
VariableName3 Type Value
When you instantiate an object from the class, you also set the value for each field in the object.

Figure 2: *Instance or Object of a Class*

3. ***Fields -*** *Fields, also known as attributes or properties, are the data members of a class. They hold the data for an object and define its state. In OOP, fields can have different levels of visibility (private, protected, or public) to control access to the data. This encapsulation protects the integrity of the object's data.*

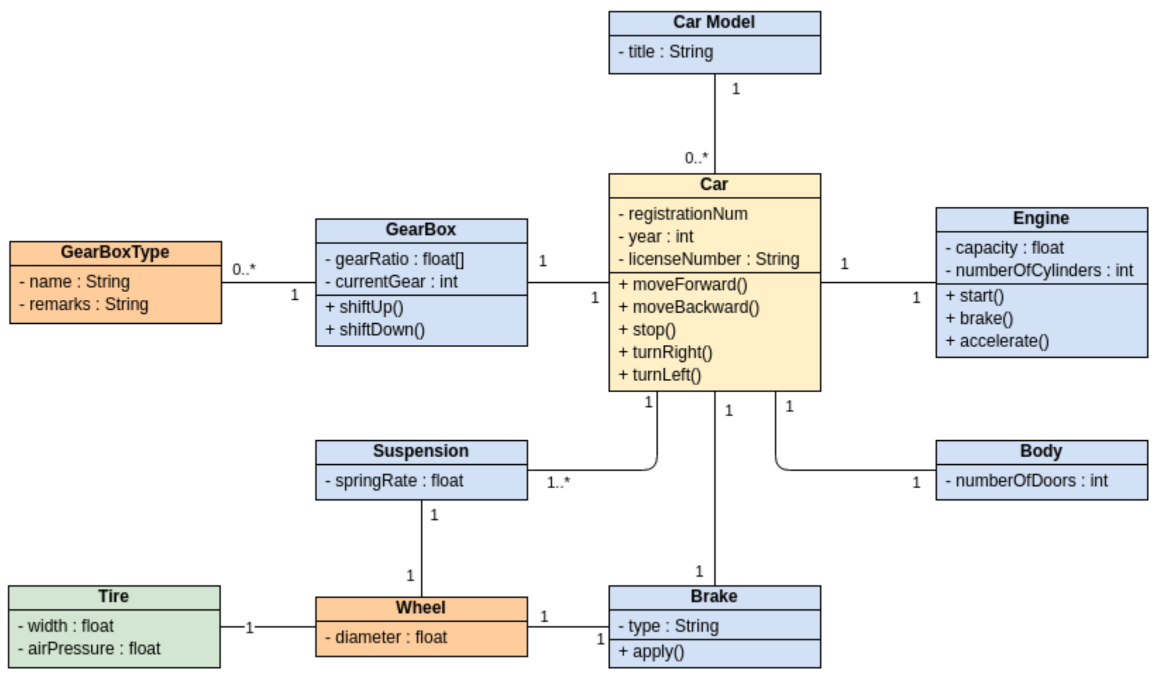


Figure 3: ***Car Features***

The image depicts a basic "Car" class in object-oriented programming. The "Car" class has three key features or "fields": its "color," "model," and current "speed." These fields define the specific characteristics of each individual car object. Encapsulation, a fundamental principle in object-oriented programming, is like a protective shield around these fields. It restricts direct access to these internal details, ensuring the car's data remains safe and the car itself functions reliably.

4. ***Methods*** *- Methods are functions defined inside a class that describe the behaviors or actions that an object of the class can perform. Methods can access and modify the data*

of the object they belong to. They allow objects to exhibit functionality that is consistent with their real-world counterparts.

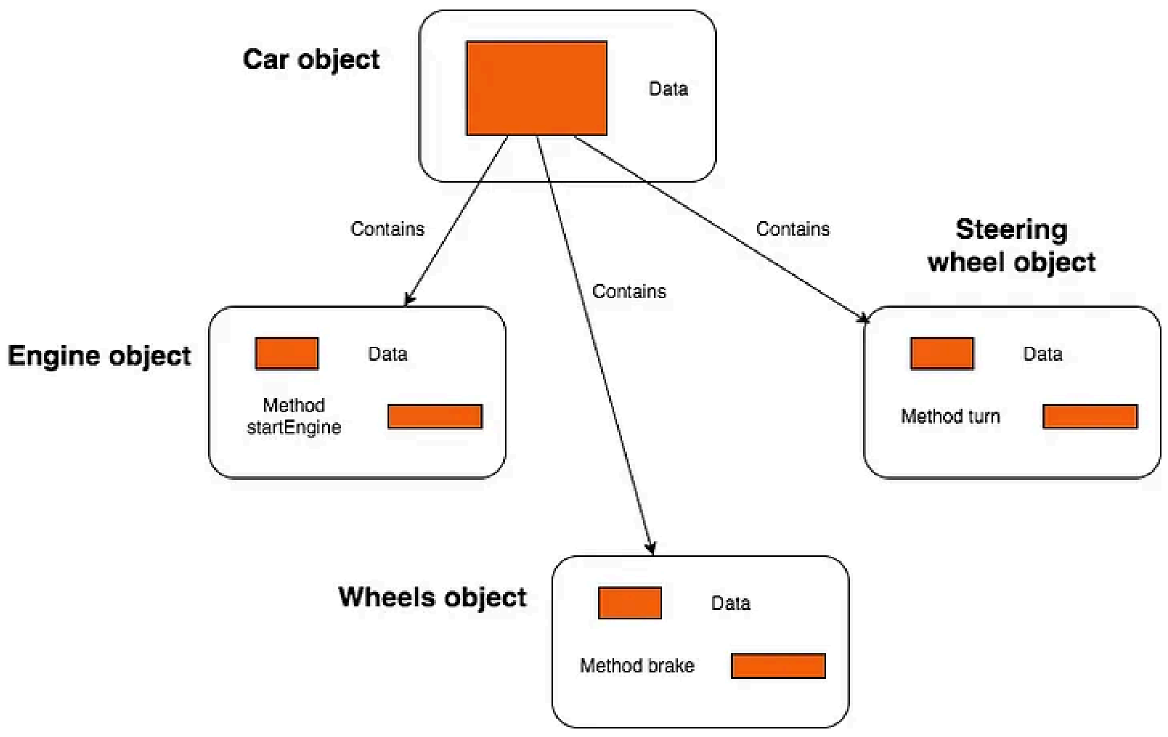
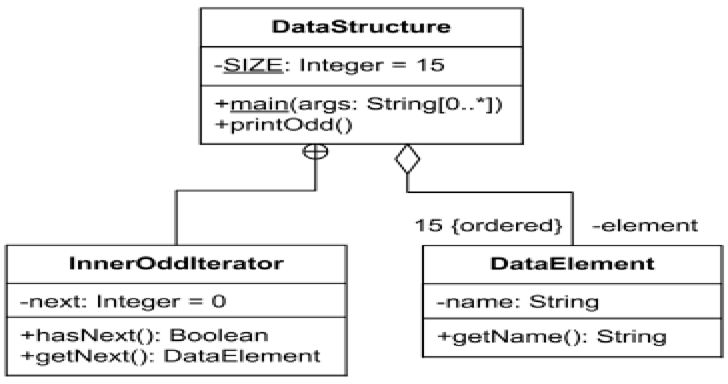


Figure 4: *Car Methods*

This image shows a "Car" class, like a blueprint for creating car objects. The "methods" are like the car's actions: "startEngine()" to turn on the engine, "accelerate()" to speed up, and "brake()" to slow down. These methods allow the "Car" objects to behave like real cars, making them more than just a set of features.

5. ***Properties -*** *Properties in OOP provide a mechanism to control access to a class's fields. They use getter and setter methods to encapsulate the field and ensure that the data can be accessed or modified in a controlled way. This allows for validation, logging, or other processing when a field's value is accessed or changed.*



The image illustrates the concept of "Properties" in object-oriented programming. A "Product" class has a "price," but this price is kept "private" (hidden). To access or change the price, you use a "Property" – like a controlled gate. This gate ensures the price is valid (not negative), allows you to track price changes, and even convert the price to different currencies, making the code more secure and easier to maintain.

III. Results

In this laboratory activity, we explored the fundamental concepts of Object-Oriented Programming (OOP), focusing on the creation and utilization of Classes, Objects, Fields, Methods, and Properties. The first step was defining a `Car` class, which served as a blueprint for creating objects with attributes like `make`, `model`, and `year`. A method named `display_info` was included to output the details of the car object. After the class definition, objects were created as instances of the `Car` class. For example, `car1` and `car2` represented distinct cars with unique data, demonstrating how classes can be used to instantiate objects with individual attributes.

Fields were defined within the class to hold data related to the objects, such as the `make`, `model`, and `year` of the car. These fields were encapsulated within the class, ensuring that the internal data could only be accessed or modified through the object's methods or properties. Methods such as `display_info` allowed the objects to display their details, illustrating how objects can exhibit behaviors as defined in the class. Additionally, properties were introduced to control access to the fields, allowing for validation and encapsulation of data, ensuring that only valid values could be assigned to certain fields.

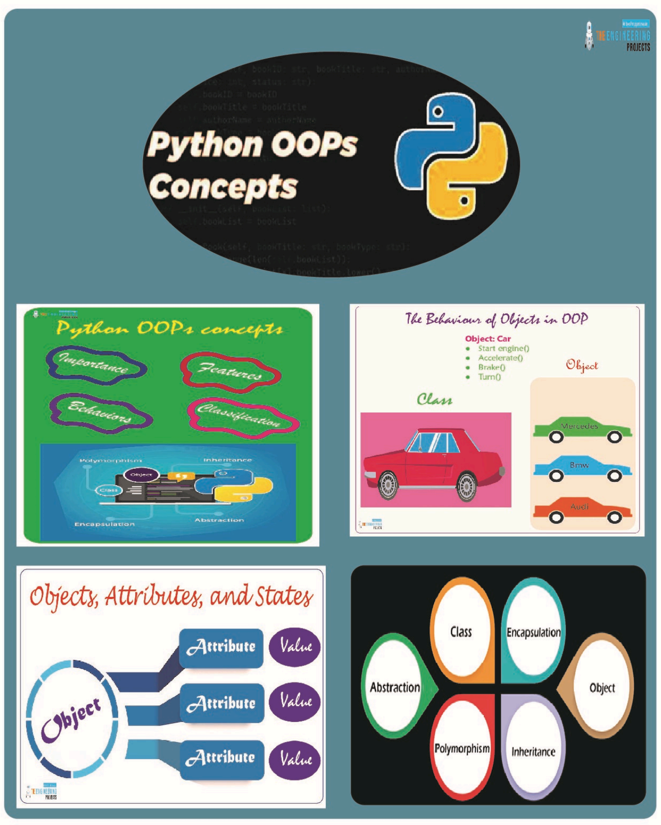


Figure 7: Python OOPs Concepts

For reference refer to this link

What is object oriented programming (with Example)

IV. Conclusion

Through this laboratory, we have explored key Object-Oriented Programming concepts such as classes, objects, fields, methods, and properties. These concepts provide a structured way of programming that models real-world entities, making the code more modular and easier to maintain. OOP also facilitates reusability through inheritance and polymorphism, enhancing the efficiency of software development. The comparison with other paradigms highlights OOP's ability to handle complex systems more effectively by promoting a clear separation of concerns and encapsulation.

Reference

"Built-in functions," Python Documentation. https://docs.python.org/3/library/functions.html#property

D. Amos, "Object-Oriented Programming (OOP) in Python," Dec. 15, 2024. https://realpython.com/python3-object-oriented-programming/#objects

L. P. Ramos, "Python's property(): Add Managed Attributes to Your Classes," Dec. 15, 2024. https://realpython.com/python-property/

"Python - classes and objects." https://www.tutorialspoint.com/python/python_classes_objects.htm

"Python Standard Library Functions | Programiz." https://www.programiz.com/python-programming/methods

"W3Schools.com." https://www.w3schools.com/python/python_classes.asp