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:
Maringal, Czer Justine D.
SAT: 12pm-8:30pm/1-A

Submitted to
Engr. Maria Rizette 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:

A. Define and discuss the following Object-oriented programming concepts:

1. Classes
2. Objects
3. Fields
4. Methods
5. Properties

Classes

A class is like a blueprint for an object. It represents the set of properties or methods that are common to all objects of one type. A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class. Represent attribute and purpose.

For Example: Consider the Class of Car. There may be many Cars with different names and brands, model, year, etc. So here, cars are the class, and brands, model, year, etc. are their properties.

Objects

Anything that has attributes and a purpose. It is a basic unit of Object-Oriented Programming and represents real-life entities. An Object is an instance of a Class. An object has an identity, state, and behavior. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

For example, “Dog” is a real-life Object, which has some characteristics like color, Breed, Bark, Sleep, and Eats.

Fields (Attributes)

Fields, also known as attributes or properties, are variables that are defined within a class. They represent the state or data of a class. Each object of the class will have its own copy of the fields. A language-specific term for instance variable, that is an attribute whose value is specific to each object.

Methods

Methods are functions defined inside a class that describe the behaviors of the objects created from the class. Methods can manipulate the object's fields and provide functionality. A language-specific term for instance variable, that is, an attribute whose value is specific to each object.

Properties

Properties provide a way to control access to the attributes of a class. They allow you to define methods to get and set the value of an attribute, providing encapsulation. A broad concept used to denote a particular characteristic of a class, encompassing both its attributes and its relationships to other classes.

III. Results

Class

```python
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year

    def display_info(self):
        print(f"{self.year} {self.brand} {self.model}")

# Car is a class
```

We all see that in my example, car is the class defines the properties and behaviors that its objects will exhibit. and brand, model, year are the properties.

Objects

```
my_car = Car("Toyota", "Corolla", 2020)
my_car.display_info()  # Output: 2020 Toyota Corolla

# my_car is an object of the Car class
```

On this picture, we can see that my_car is an example object of the Car class that have specific values for the attributes defined in the class.

Fields (Attributes)

```
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand  # brand is a field
        self.model = model  # model is a field
        self.year = year    # year is a field
```

Now, the brands, models, and years is a fields or attributes of the Car Class that hold data specific to an object.

Methods

```
class Car:
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year

    def display_info(self):
        print(f"{self.year} {self.brand} {self.model}")

# display_info is a method of the Car class
```

The display_info is a method of the Car Class that describe the behaviors or actions that an object can perform.

Methods

```
class Car:
    def __init__(self, brand, model, year):
        self._brand = brand  # Using underscore to indicate a private attribute
        self._model = model
        self._year = year

    @property
    def brand(self):
        return self._brand

    @brand.setter
    def brand(self, value):
        self._brand = value

    def display_info(self):
        print(f"{self._year} {self._brand} {self._model}")

# my_car.brand is a property that controls access to the _brand attribute
my_car = Car("Toyota", "Corolla", 2020)
print(my_car.brand)  # Output: Toyota
my_car.brand = "Honda"
print(my_car.brand)  # Output: Honda
```

The picture shows the result of all OOP concepts given that emphasize the properties provide a way to control access to the attributes of a class. It defines methods to get and set the value of an attribute.

IV. **Conclusion**

In Object-Oriented Programming (OOP) the concepts of classes, object fields, methods, and properties are interdependent and work together to form a coherent system. A class acts like a blueprint defining the structure, i.e., fields of its objects and the behaviors, i.e., methodology of its objects. An object represents an instance of a class, embodying a tangible and specific real-world entity that possesses fields assigned certain values and is capable of executing particular actions due to its methods. Fields are responsible for storing the data or state of an object, whereas methods articulate the actions or behaviors that an object can undertake, thereby interacting with or altering its state. Properties serve as intermediaries, ensuring encapsulation and data integrity by facilitating controlled access to fields via getter and setter methods, which are crucial for secure accessibility. However, these components work in unison to promote modularity, reusability and maintainability; the definitions of each class within the system can be distinctly separated from their corresponding instances (objects), all the while safeguarding and managing state and behavior effectively.

Reference

Website

https://www.youtube.com/watch?v=6m0DZSfZ5AQ&t=64s

https://www.scholarhat.com/tutorial/python/oops-concepts-in-python

https://analyticsdrift.com/python-oops/

https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/