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:
Nerio, Hannah Grace A.
Sat (4:30 to 8:30) / BSCpE 1-A

Submitted to
Engr. Maria Rizette Sayo
Intructor

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

It is a collection of data and member functions that manipulate data. The data components of class are called data members and functions that manipulate the data are called member functions. It can also be called blueprint or prototype that defines the variables and functions common to all objects of certain kind. It is also known as user defined data type or ADT(abstract data type) A class is declared by the keyword class.

```
class MyClass
{
        private int value;

        MyClass(int n)
        {  value = n;  }

        void perform(int m)
        {       for (int k=1; k<=value; k++)
                    System.out.print (m*k + "  ");
        }

        int compute()
        {  return value*value;  }
}
```

2. Objects

Object is a class variable or an instance of class. It can represent a person, a bank account or any item that a program can handle. When a program is executed, the objects interact by sending messages to one another. For example, if ‘customer’ and ‘account’ are two objects in a program, then the customer object may send message to account object requesting for a bank balance. Each object contains data and code to manipulate 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 the type of response returned by the objects.

```
MyClass first = new MyClass(5);
MyClass second = new MyClass(3);

first.perform(6);
Prints:     6  12  18  24  30

second.perform(-4);
Prints:     -4  -8  -12
```

3. Fields

   A field is a member variable that is declared within a class or struct and is used to hold data for an instance of a class/objects of a class. All non-static fields are called “instance fields”. As a naming rule, all private fields must have “_” before their name and must start with a not capital letter. All Fields are typically declared at the class level, and are used to represent the state of an instance of a class and can be accessed by methods. (And properties of a class.)

```
public class PersonTitleName
{
    //Fields
    private string _name;
    private string _title;

    //...

}
```

   Fields let you separate data from the other code parts outside of the class by having necessary access modifiers like protected, private, and public and encapsulating it within a class. This can improve the organization, maintainability, and error prevention of your code.

```
public class PersonTitleName
{
    //Fields
    private string _name;
    private string _title;

    //initializing fields with a constructor
    public PersonTitleName(string Name, string Title)
    {
        this.Name = Name;
        this.Title = Title;
    }

    public string GetName()
    {
        return Name;
    }

    public string SetName(string Name)
    {
        this.Name = Name;
    }

    public string GetTitle()
    {
        return Title;
    }

    public string SetTitle(string Title)
    {
        this.Title = Title;
    }
    public void EnhanceTitle(string title2)
    {
        Title += title2;
    }

    public string GetFullName()
    {
        return Title + Name;
    }
}
```

4. Methods

   Methods are the action blocks that executes an operation as per user direction with/without input parameters. Similarly, methods can be broadly classified into three types: *Standard method, Class method and Static method* (explained using standalone example). Instance methods are the most common type of methods in Python classes. They are associated with instances of a class and have access to

```
class Dog:
    def __init__(self, name, breed):
        self.name = name
        self.breed = breed

    def bark(self):
        return f"{self.name} is barking!"
```

both instance-specific data (attributes) and class-level data. Here's an example of an instance method:

Static methods are defined using the @staticmethod decorator. They do not have access to the instance or class state and are not bound to either. Static methods behave like regular functions but are scoped within the class for organization. Here's an example:

```
class MathUtils:
    @staticmethod
    def add(x, y):
        return x + y
```

Class methods are defined using the @classmethod decorator. They are bound to the class and can access and modify class-level attributes. Class methods take the class itself as their first argument, typically named cls. Here's an example:

```
class MyClass:
    count = 0  # Class-level attribute

    def __init__(self, data):
        self.data = data
        MyClass.count += 1

    @classmethod
    def get_count(cls):
        return cls.count
```

5. Properties

   The properties of Object-Oriented Programming (OOP) refer to the core principles or features that define the paradigm and guide how objects and classes interact within the system. These properties are fundamental to the design and structure of OOP-based systems. The four main properties are:

   Data abstraction is one of the most essential and important features of object-oriented programming. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car, but he does not know about how on pressing the accelerator the speed is increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc in the car. This is what abstraction is.

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. In Encapsulation, the variables or data of a class are hidden from any other class and can be accessed only through any member function of their class in which they are declared. As in encapsulation, the data in a class is hidden from other classes, so it is also known as data-hiding.

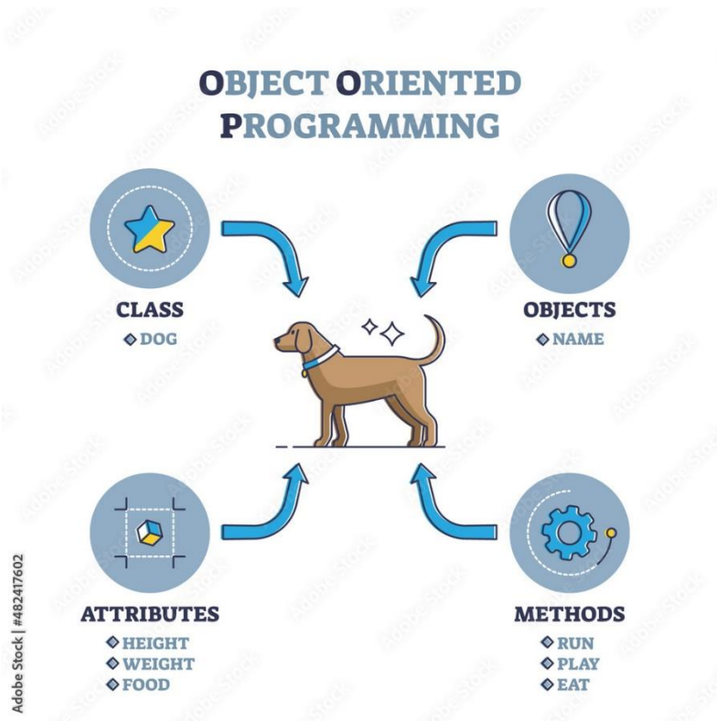
Inheritance is an important pillar of OOP(Object-Oriented Programming). The capability of a class to derive properties and characteristics from another class is called Inheritance. When we write a class, we inherit properties from other classes. So when we create a class, we do not need to write all the properties and functions again and again, as these can be inherited from another class that possesses it. Inheritance allows the user to reuse the code whenever possible and reduce its redundancy.

The word **polymorphism** means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. For example, A person at the same time can have different characteristics. Like a man at the same time is a father, a husband, an employee. So the same person posses different behavior in different situations. This is called polymorphism.

Example of inheritance: https://media.geeksforgeeks.org/wp-content/uploads/20200911171738/InheritanceinObjectOrientedProgramming.png

Example of Polymorphism: https://media.geeksforgeeks.org/wp-content/uploads/20200911171857/PolymorphisminObjectOrientedProgramming.png

III. Results



The picture below explains the basic concepts of Object-Oriented Programming (OOP) using a dog as an example. A class is like a blueprint that defines the structure and behavior of objects. In this case, the class is “Dog,” which describes what characteristics and actions all dogs can have. An object is a specific instance of a class, such as a dog named “Buddy.” Objects are created based on the blueprint provided by the class. Each object has attributes, which are the details or properties that describe it. For example, a dog’s attributes might include its height, weight, and food preferences. Additionally, objects can perform methods, which are actions or behaviors defined in the class. For a dog, these methods could include running, playing, or eating. Together, these elements form the foundation of OOP, allowing developers to create reusable, organized, and efficient code by modeling real-world entities and their behaviors.

https://youtu.be/JeznW_7DlB0?si=arrLCxiPmoI0TdW2

This youtube video will tackle more about the object oriented programming

IV. Conclusion

In conclusion, this laboratory activity provided valuable insights into the practical application of Object-Oriented Programming (OOP) principles. It demonstrated how key concepts such as classes, objects, inheritance, and polymorphism are implemented to *create more organized and reusable code.* By completing the tasks, we were able to see firsthand how OOP enhances the efficiency and maintainability of software development. The activity not only solidified our understanding of theoretical concepts but also helped us gain practical experience in applying them to real-world programming scenarios.

Reference

Adobe Stock, "Object-Oriented Programming Language (OOP) Paradigm Explanation Outline Diagram," *Adobe Stock*, [Online]. Available: https://stock.adobe.com/ch_fr/images/object-oriented-programming-language-or-oop-paradigm-explanation-outline-diagram-labeled-educational-scheme-with-class-objects-attributes-and-methods-for-coding-system-and-type-vector-illustration/482417602. [Accessed: 18-Jan-2025].

Avijit Bhattacharjee, "Instance Methods, Static Methods, and Class Methods in Python: A Comprehensive Guide," *Medium*, [Online]. Available: https://medium.com/@avijit.bhattacharjee1996/instance-methods-static-methods-and-class-methods-in-python-a-comprehensive-guide-9163de3f4a47?fbclid=IwZXh0bgNhZW0CMTEAAR2u84I0bBd8HppnJvBnq1JcpKqMzt7XA5BDwMQnlnaNV-ArmbELY_aPaS0_aem_oQ6Hl6XAV8yucFqnpqsfSQ. [Accessed: 18-Jan-2025].

Cem Tuganli, "Difference Between Field and Property in C# OOP," *Medium*, [Online]. Available: https://medium.com/@cemtuganli/difference-between-field-and-property-in-c-oop-b0ced5b381c0?fbclid=IwZXh0bgNhZW0CMTEAAR0Mp2FydXqks-Qn4wKo4AZLcBfGFU122uzNvzn6JJ7JXWGmEYuXAEbIyPY_aem_o_BuO8jNTJhICheooEG7Ew. [Accessed: 18-Jan-2025].

GeeksforGeeks, "Introduction of Object-Oriented Programming," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/?fbclid=IwZXh0bgNhZW0CMTEAAR3BNRG-i4pp2eeWzuZYMpllCr0a55BYFTEZooAI89j1vfnnadf02TiG4qc_aem_KU8ykZT1601ki65KYCITXw. [Accessed: 18-Jan-2025].

Maturi R. College of Engineering and Technology, "Object-Oriented Programming," *Maturi R. College of Engineering and Technology*, [Online]. Available: https://mrcet.com/downloads/digital_notes/HS/OOP_10122018.pdf. [Accessed: 18-Jan-2025].
National Institute of Open Schooling, "Lesson 13 - Object-Oriented Programming," *National Institute of Open Schooling*, [Online]. Available: https://www.nios.ac.in/media/documents/330srsec/online_course_material_330/Theory/Lesson_13.pdf. [Accessed: 18-Jan-2025].

S. Selvakumar, "101 - Introduction to Class, Attributes, Objects, Methods," *LinkedIn*, [Online]. Available: https://www.linkedin.com/pulse/101-introduction-class-attributes-objects-methods-selvakumar?fbclid=IwZXh0bgNhZW0CMTEAAR2t5_nzMgL5mDJfQqP4QJohCcdfBhVkmadwkQynKwKSRZsCQzdq8MiRS7A_aem_jxAD4EWBEVcNh70xwqslHw. [Accessed: 18-Jan-2025].