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:
Disomnong, Jalilah M.
Saturday 12:00PM-4:00PM 4:30PM-8:30PM/ BSCpE-1A

Submitted to
Engr. Maria Rizette H. 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

   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. It represents the set of properties or methods that are common to all objects of one type. A class is like a blueprint for an object.

   For Example: Consider the Class of Cars. There may be many cars with different names and brands but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range, etc. So here, Car is the class, and wheels, speed limits, mileage are their properties.

2. Objects

   It is a basic unit of Object-Oriented Programming and represents the real-life entities. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. 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.

3. Fields

Some OOP concepts are not common to all languages or were introduced later as languages evolved. We mention a few of these:

Multiple Inheritance: This is when a subclass inherits from multiple base classes. C++ and Python support this. In Java and C#, a class can implement multiple interfaces but not inherit from multiple classes.

Generics: Classes, methods and interfaces that can be parameterized are called generics. For example, a class might implement a set of objects with parameterized type. Thus, the same class definition can be used to instantiate a set of Shape objects or Car objects.

Templates: Whereas types in generics are substituted at runtime, types in templates are specialized at compile time. Each template specialization has its own compiled code. C++ and D support templates.

Delegates: We can call a method via its delegate that has a compatible method signature. With delegates, methods can be passed as arguments to other methods.

Reflection: Not exclusive to OOP, reflection allows a class to introspect and modify its own definition. In Java, reflection is possible on classes, interfaces, fields and methods at runtime. This is often useful for testing, data serialization and metaprogramming.

4. Methods

   In object-oriented programming (OOP), a method is a programmed procedure that is defined as part of a class and is available to any object instantiated from that class. Each object can call the method, which runs within the context of the object that calls it. This makes it possible to reuse the method in multiple objects that have been instantiated from the same class.

5. Properties

   Object-oriented programming has four basic concepts: encapsulation, abstraction, inheritance and polymorphism.

1. Encapsulation hides a class's internal data by bundling it with its methods, exposing only necessary functions and protecting it from direct access.

2. Abstraction simplifies complex code by separating a class's interface from its implementation, allowing easier modification without affecting external code.

3. Inheritance allows a new class to inherit properties and methods from a parent class, enabling code reuse and the creation of class hierarchies.

4. Polymorphism allows objects of different classes in a hierarchy to be treated uniformly, with behaviors that may vary depending on the object type.

III. Results

1. Classes

   A class is considered a blueprint of objects.

   We use the class keyword to create a class in Python. For example,

```
class ClassName:
    # class definition
```

Figure 1 https://www.programiz.com/python-programming/class

Here, we have created a class named ClassName.

Let's see an example,

```
class Bike:
    name = ""
    gear = 0
```

Figure 2 https://www.programiz.com/python-programming/class

Here,

- Bike - the name of the class
- name/gear - variables inside the class with default values "" and **0** respectively.

2. Objects

   An object is called an instance of a class.

Suppose Bike is a class then we can create objects like bike1, bike2, etc from the class.

Here's the syntax to create an object

```
objectName = ClassName()
```

Figure 3 https://www.programiz.com/python-programming/class

```
# create class
class Bike:
    name = ""
    gear = 0

# create objects of class
bike1 = Bike()
```

Figure 4 https://www.programiz.com/python-programming/class

Example:

Here, bike1 is the object of the class. Now, we can use this object to access the class attributes.

3. Attributes

We use the . notation to access the attributes of a class. For example,

```
# modify the name property
bike1.name = "Mountain Bike"

# access the gear property
bike1.gear
```

Figure 5 https://www.programiz.com/python-programming/class

Here, we have used bike1.name and bike1.gear to change and access the value of name and gear attributes, respectively.

4. Methods

We can also define a function inside a Python class. A Python function defined inside a class is called a **method**.

Example:

```
# create a class
class Room:
    length = 0.0
    breadth = 0.0

    # method to calculate area
    def calculate_area(self):
        print("Area of Room =", self.length * self.breadth)

# create object of Room class
study_room = Room()

# assign values to all the properties
study_room.length = 42.5
study_room.breadth = 30.8

# access method inside class
study_room.calculate_area()
```

Run Code »

Figure 6 https://www.programiz.com/python-programming/class

Output

```
Area of Room = 1309.0
```

Figure 7 https://www.programiz.com/python-programming/class

In the above example, we have created a class named Room with:

Attributes: length and breadth

Method: calculate_area()

Here, we have created an object named study_room from the Room class.

We then used the object to assign values to attributes: length and breadth.

Notice that we have also used the object to call the method inside the class,

```
study_room.calculate_area()
```

Figure 8 https://www.programiz.com/python-programming/class

Here, we have used the . notation to call the method. Finally, the statement inside the method is executed.

5. Properties

Take the following Movie class as an example; it takes a director, cost of production and revenue gained from the project.

```
class Movie:
    def __init__(self, director, cost, revenue):
        self.director = director
        self.cost = cost
        self.revenue = revenue
```

Figure 9 https://www.codearmo.com/python-tutorial/properties#

The property decorator denoted by @property is the most common way to declare a decorator in Python. The protocol for implementing a property that has read and write access is to include both a getter and a setter.

IV. Conclusion

Object-Oriented Programming (OOP) helps organize code into reusable and easy-to-maintain pieces. It uses concepts like classes, objects, inheritance, and polymorphism to model real-world things in code. In Python, OOP makes it easier to create flexible and scalable software by organizing it into classes and objects, which can interact with each other.

OOP also encourages reusing code, reducing redundancy, and protecting data through encapsulation and abstraction. This makes it simpler to extend and modify software without affecting other parts. Overall, OOP helps create clean, efficient, and maintainable code for modern applications.

Reference

Book

[1]

Website

[1] Devopedia. 2022. "Object-Oriented Programming Concepts." Version 5, February 15. Accessed 2024-06-25. https://devopedia.org/object-oriented-programming-concepts

[2] GeeksforGeeks, “Introduction of object oriented programming,” GeeksforGeeks, Feb. 09, 2023. https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/

[3] GeeksforGeeks, “Python classes and objects,” GeeksforGeeks, Dec. 13, 2024. https://www.geeksforgeeks.org/python-classes-and-objects/

[4]Herrity, K. (2024) A complete guide | indeed.com, What Is Object-Oriented Programming (OOP)? A Complete Guide. Available at: https://www.indeed.com/career-advice/career-development/what-is-object-oriented-programming.

[6] “Properties in Python classes.” https://www.codearmo.com/python-tutorial/properties#