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:
ACEBEDO, SEBASTIAN C.
SAT-4:30-8:30 / BSCPE 1A

Submitted to
ENGR. MARIA RIZETTE SAYO
Professor/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 OOP 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:

Object-Oriented Programming (OOP) is the process of turning a design into functional software that fullfills the initial request of the product owner. Additionally, Programs can be structured using the object-oriented programming paradigm, which bundles properties and behaviors into separate objects.

1. **Classes**
   - In OOP, a class is the template or blueprint that provides the general description of a set of objects. It defines the properties, attribute, and the behaviors of methods that objects belonging to that class will have.
2. **Objects**
   - An object is an instance of a class. It's a concrete realization of the class blueprint. One actual entity with its own unique set of values for the properties defined in the class.
3. **Fields**
   - Fields, also known as attributes or properties, are variables that hold values regarding an object. They define the characteristics or properties of an object.
4. **Methods**
   - Methods are functions that define the behavior or actions that an object can perform.
5. **Properties**

- Properties offer a controlled and flexible way to access and modify the field values of an object.

III. Results

Many items of the same kind are frequently found in the actual world. Your bicycle, for instance, is merely one of many bicycles worldwide. We define your bicycle object as an instance (in the glossary) of the class of objects called bicycles using object-oriented language. Both state (current gear, current cadence, two wheels) and behavior (change gears, brake) are shared by bicycles. The condition of each bicycle, however, is distinct from and independent of that of other bicycles.

Bicycle manufacturers use the fact that bicycles have many qualities in common to create a large number of bicycles from a single plan. It would be extremely inefficient to create a fresh blueprint for each bicycle that is made.

A class can be depicted as shown below:

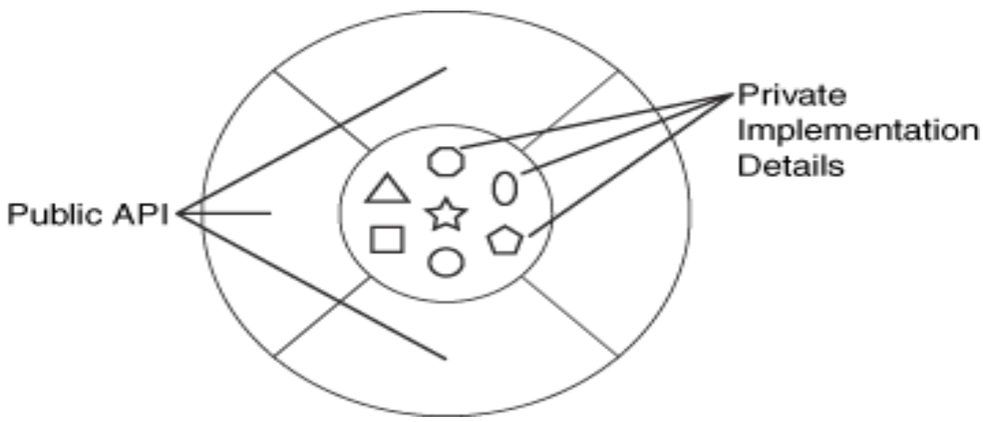

Figure 1.1: Bicycle diagram

Source: What Is a Class?

The instance variables required to hold the current gear, the current cadence, and so on for each bicycle object would be declared by the class for our bicycle example. As seen in the following picture, the class would also declare and supply implementations for the instance methods that let the rider brake, shift gears, and alter the pedaling cadence.

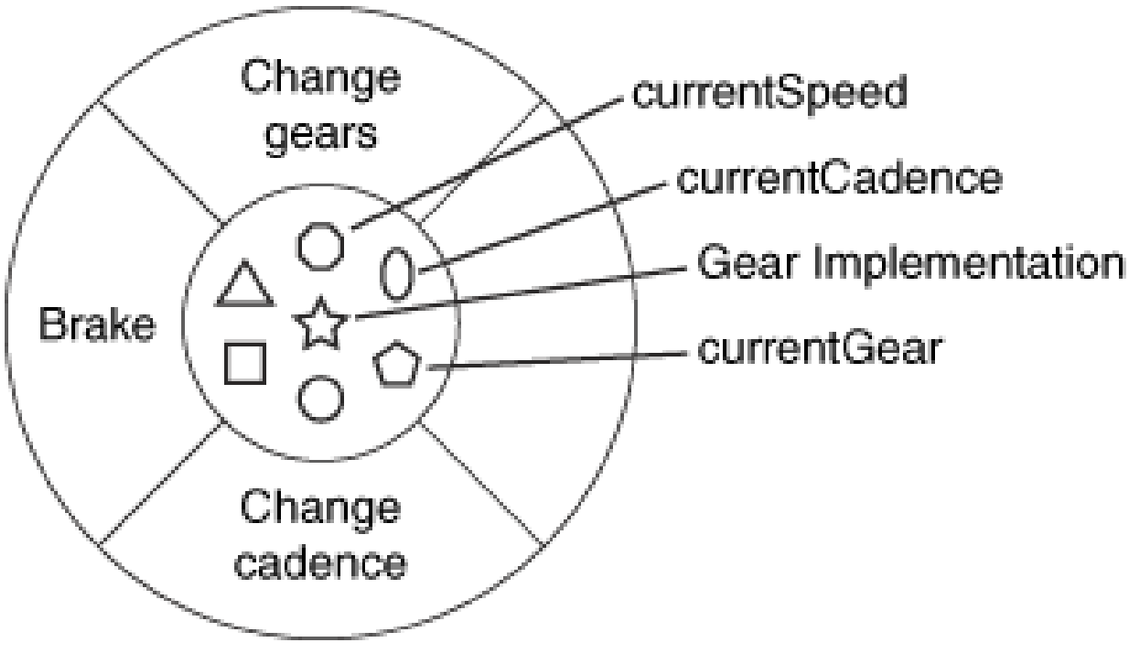

Figure 1.2: Bicycle diagram

Source: What Is a Class?

Any number of bicycle objects can be produced from the bicycle class once it has been formed. The system allots sufficient memory for the object and all of its instance variables when you create an instance of a class. As seen in the following graphic, every instance receives a copy of every instance variable defined in the class.

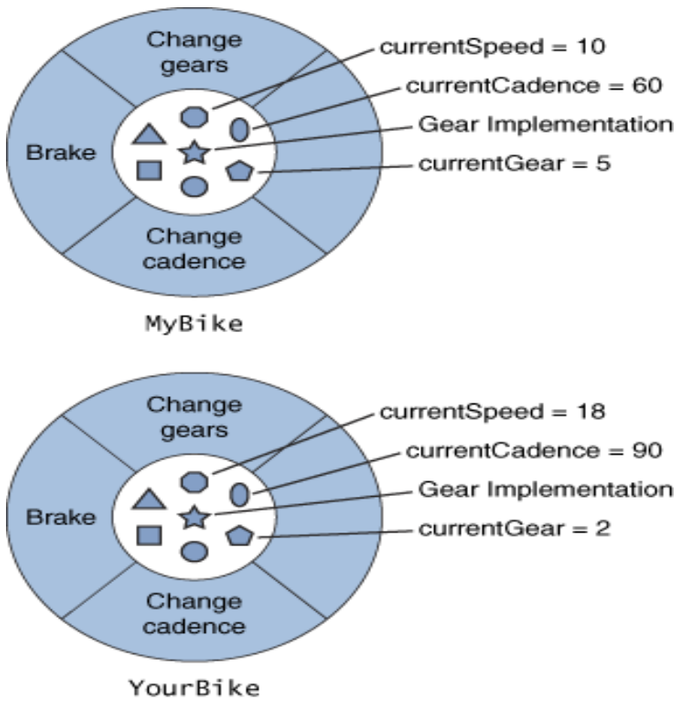

Figure 1.3: Bicycle diagram

Source: What Is a Class?

All instances of the class share the information contained in the class variable. Let's say, for instance, that every bicycle has the same amount of gears. It is inefficient to define an instance variable to store the number of gears in this situation because each instance would have a duplicate of the variable, but the value would be the same. In these cases, the number of gears can be defined as a class variable. This variable is shared by all instances. All other objects of that kind will also change if one object modifies the variable. According to the glossary, a class may also declare class methods. While instance methods must be invoked on a specific instance, class methods can be called directly from the class.

IV. Conclusion

In object-oriented programming, classes act as templates that outline the form and functions of objects. Objects are specific instances of these classes, each having its own state kept in fields and distinct behavior carried out through methods. Properties connect fields and methods, offering regulated access to an object's internal data. The bicycle example demonstrates the effectiveness and modularity of this method, where common attributes and behaviors, including gears, cadence, and speed, are contained within the class definition. Employing instance variables for specific states and class variables for shared information promotes efficient memory use and maintains consistency. Additionally, methods enable dynamic interactions, such as changing gears or applying brakes, showcasing the flexibility of object-oriented programming. This approach encourages reusability, scalability, and clarity in software architecture, which makes it suitable for representing real-world entities and addressing intricate issues in a systematic and effective way.

Reference

Book

"Python Object-Oriented Programming," *Google Books*.
https://books.google.com.ph/books?hl=en&lr=&id=J5Y2EAAAQBAJ&oi=fnd&pg=PP1&dq=classes+and+object+in+python&ots=j77fJOzc8Z&sig=I9_Pe-KubmAlaSTw-ujJtJOw7DY&redir_esc=y#v=onepage&q=classes%20and%20object%20in%20python&f=false

Website

J. Crotts, "Object-Oriented Programming," in *Learning Java*, 2024, pp. 191–323. doi: 10.1007/978-3-031-66638-4_4.

"What is a class?"
https://www.iitk.ac.in/esc101/05Aug/tutorial/java/concepts/class.html#:~:text=Definition%3A%20A%20class%20is%20a,objects%20of%20a%20certain%20kind.

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