## 

## 

**Experiment 3:**

**Object Oriented Design and Implementation**

CPE106L (Software Design Laboratory)

**Member 1: Claros, Angelica A.**

**Member 2: Facal, Ma. Catherina N.**

**Member 3: Santos, Angelica Anne D.**

Group No.: **5**

Section: **B2**

## **PreLab**



|  |
| --- |
| **Laboratory Insights** |

**Claros**

In the third laboratory, we delved deeper into understanding the functions involved in the Python programming language. After understanding data structures in Python, we progressed to understanding classes and modules essential for coding longer or more complex instances, such as building software applications.

Through the utilization of the book available in METIS, I grasped the concept and function of classes and modules, as well as their importance. Essentially, classes serve as blueprints containing the functions and data of specific objects. Also, with classes, it becomes efficient to understand the functions of available objects as the data of a class is bundled together. On the other hand, modules are broad, containing executable statements, variables, and classes – essentially, they are sets of functions that can be utilized in the current Python file being worked on. Specifically, these modules are files consisting of Python code that can be considered like a code library, thereby saving time in coding as some codes are already completed and these pre-made modules can be imported into the current Python file.

Essentially, this laboratory taught me how to efficiently and effectively code programs using classes and modules. Additionally, we were introduced to UMLet – a tool for drawing diagrams to illustrate how the code works. Using this tool, I was able to further organize and visualize the flow of my codes by visualizing the relationships between the classes, serving as a blueprint for my coding programs. Overall, I aim to further enhance my skills in using classes, expand my knowledge on various modules available in Python libraries, and use UMLet to visualize the flow of the code, thereby making my codes more understandable and saving time

**Facal**

For the third lab activity, we tackled the object-oriented design and implementation in Python. Similar to our CPE103L, similar topics such as proper code structure in a way that is more optimized and manageable. This can be through the use of classes, modules, etc. Classes are Python data structures that can be used as a template to make codes more structured. Meanwhile, modules are existing Python files that can be imported to our codes. In addition, we were introduced to UMLet and created UML diagrams. Unified Modified Language (UML) is used primarily as documentation for software programs that provides a graphical representation for a program, while UMLet is a free, easy and open-source tool in creating UML diagrams. This can be used for future projects to better represent and comprehend software programs.

Upon reviewing and analyzing the provided PreLab materials, I gained a deeper comprehension of the syntax used with use-case diagrams, including elements such as actors, use cases, subject boundaries, association relationships, include relationships, extend relationships, and generalization relationships. Furthermore, I familiarized myself with the essential steps involved in constructing a use-case diagram, including the placement of use cases, actors, drawing subject boundaries, and incorporating associations.

**Santos**

This Prelab Number 3 focused on object-oriented design and implementation in Python which mainly dealt with classes, modules, and methods. Classes acts as templates for creating objects which represent real-world entities. While module contain code that can be imported into other programs and methods, facilitate actions on objects.

To dig deeper, using the book I understood the use and functionality of classes and modules that work hand in hand. As said earlier, classes are like templates for creating objects, defining what they can do and what information they contain. Modules are collections of ready-to-use code that we can plug into our programs to save time. Together, they help us write better Python code faster. Moreover, the lab covered important concepts like use case and class diagrams, which help visualize software systems using UML. Despite some initial difficulties, UMLet, made it easier to create these diagrams, showing how classes and objects relate to each other. I also learned about inheritance, aggregation, and composition, which are key principles in object-oriented design.

The lab gave us a better understanding of object-oriented programming in Python. Through classes, modules, and methods, we learned how to organize and reuse code effectively. Using UML diagrams helped visualize our designs, enhancing our grasp of software development techniques.

**Answers to Questions**

1. a. Is owned by a particular instance of a class and no other

2. c. self

3. b. set the instance variables to initial values

4. b. always must have at least one parameter name, called self

5. b. the entire class in which it is introduced

6. b. when it can no longer be referenced anywhere in a program

7. a. all instances of a class have in common

8. b. A.\_\_init\_\_(self)

9. b. pickle them using the pickle function dump

10. a. has a single header but different bodies in different classes