| **Laboratory Activity No. 3.1** | |
| --- | --- |
| **Introduction to Object-Oriented Programming** | |
| **Course Code:** CPE103 | **Program:** BSCPE |
| **Course Title:** Object-Oriented Programming | **Date Performed:** 01/25/25 |
| **Section:** BSCpE 1A | **Date Submitted:**01/31/25 |
| **Name:** Nerio, Hannah Grace A. | **Instructor:** Engr. Maria Rizzete Sayo |
| **1. Objective(s):** | |
| This activity aims to familiarize students with the concepts of Object-Oriented Programming | |
| **2. Intended Learning Outcomes (ILOs):** | |
| The students should be able to:   * 1. Identify the possible attributes and methods of a given object   2. Create a class using the Python language   3. Create and modify the instances and the attributes in the instance. | |
| **3. Discussion:** | |
| Object-Oriented Programming (OOP) is an approach to programming that views the world and systems as consisting of objects that relate and interact with each other. This involves identifying the characteristics that describe the object which are known as the Attributes of the object. Furthermore, it also deals with identifying the possible capabilities or actions that an object is able to do which are called Methods.  An object is simply composed of Attributes and Methods wherein Attributes are variables that hold the information describing the object and Methods are functions which allow the object to perform its defined capabilities/actions. A UML Class Diagram is used to formally represent the collection of Attributes and Methods.  An example is given below considering a simple banking system.  Accounts ATM  + account\_number: int + serial\_number: int  + account\_firstname: string  + account\_lastname: string  + current\_balance: float  + address: string + deposit(account: Accounts, amount: int) + email: string + widthdraw(account: Accounts, amount: int) + update\_address(new\_address: string) + check\_currentbalance(account: Accounts) + update\_email(new\_email: string) + view\_transactionsummary() | |
| **4. Materials and Equipment:** | |

| Desktop Computer with Anaconda Python/Python Colab Windows Operating System |
| --- |
| **5. Procedure:** |
| **Creating Classes**   1. Create a folder named **OOPIntro\_LastName** 2. Create a Python file inside the **OOPIntro\_LastName** folder named **Accounts.py** and copy the code shown below: |

| 1. Modify the Accounts.py and add ***self,*** before the new\_address and new\_email. 2. Create a new file named ATM.py and copy the code shown below:     **Creating Instances of Classes**   1. Create a new file named main.py and copy the code shown below: |
| --- |

6.

Run the main.py program and observe the output. Observe the variables names account\_firstname, account\_lastname as well as other variables being used in the Account1 and Account2. 7. Modify the main.py program and add the code underlined in red.

8. Modify the main.py program and add the code below line 38.

| 9. Run the main.py program.  **FOR CHECKING PLEASE REFER TO THIS LINK:** <https://github.com/HannahGraceNerio/CPE-103-OOP-1-A/blob/main/Laboratory_3.ipynb>  **Create the Constructor in each Class**   1. Modify the Accounts.py with the following code:   Reminder: def init (): is also known as the constructor class   1. Modify the main.py and change the following codes with the red line. Do not remove the other codes in the program. |
| --- |

| 3. Run the main.py program again and run the output.  **FOR CHECKING PLEASE REFER TO THIS LINK:** <https://github.com/HannahGraceNerio/CPE-103-OOP-1-A/blob/main/Laboratory_3.ipynb> |
| --- |
| **6. Supplementary Activity:** |
| **Tasks**   1. Modify the ATM.py program and add the constructor function. 2. Modify the main.py program and initialize the ATM machine with any integer serial number combination and display the serial number at the end of the program. 3. Modify the ATM.py program and add the **view\_transactionsummary()** method. The method should display all the transaction made in the ATM object.     **FOR CHECKING PLEASE REFER TO THIS LINK:** <https://github.com/HannahGraceNerio/CPE-103-OOP-1-A/blob/main/Laboratory_3.ipynb>  **Questions**   1. What is a class in Object-Oriented Programming?   In Object-Oriented Programming (OOP), a class is a blueprint for creating objects. It defines the characteristics and behaviors that the objects will have. A class acts as a template, setting rules for how objects should store information and perform actions. Instead of writing the same code repeatedly, a class allows you to create multiple objects with similar properties and behaviors.         1. Why do you think classes are being implemented in certain programs while some are sequential(line-by-line)?   Classes are used in certain programs because they help organize code, make it reusable, and allow for better structure, especially in large or complex projects. They group related data and actions together, making it easier to manage and modify. Programs that deal with objects, like managing users in a system or handling different types of vehicles, benefit from using classes.  On the other hand, some programs are written sequentially, meaning they execute line by line without using classes. If a program only performs a straightforward task, like calculating numbers or processing a simple set of instructions, a sequential approach is often more efficient and easier to understand.       1. How is it that there are variables of the same name such account\_firstname and account\_lastname that exist but have different values? |

| because those two variables that have same name are belong to a different objects or instances of a class. In Object-Oriented Programming (OOP), each object created from a class has its own copy of the attributes, means that the values are stored separately for each instance.      4. Explain the constructor functions role in initializing the attributes of the class? When does the Constructor function execute or when is the constructor function called?  The constructor function plays a key role in initializing the attributes of a class when an object is created. It ensures that every new object has the necessary attributes with their initial values.The constructor function is automatically executed when a new object is created from the class. This means you don’t need to call it manually—it runs as soon as you create an instance of the class. By using a constructor, you can set up an object’s attributes right from the start, making sure it has the necessary data before being used in the program. |
| --- |

| 5. Explain the benefits of using Constructors over initializing the variables one by one in the main program?  Using constructors instead of manually initializing variables in the main program provides several advantages. First, it automates the process, ensuring that attributes are set as soon as an object is created, which reduces repetitive code and makes the program more efficient. It also improves code organization and readability by keeping all initialization logic inside the class rather than scattering it throughout the program. |
| --- |
| **7. Conclusion:** |
| In conclusion, this laboratory activity gave me a hands-on introduction to Object-Oriented Programming (OOP), helping me understand key concepts such as classes, objects, attributes, and methods. By creating and modifying Python classes, I learned how to structure code more efficiently, making it reusable and organized. The use of constructors showed me how attributes are initialized automatically, reducing redundancy and improving maintainability. Through practical exercises, I also explored how instances of a class can have unique values while following the same blueprint. Overall, this activity emphasized the importance of OOP in building scalable and well-structured programs. |
| **8. Assessment Rubric:** |