Laboratory Activity No. 1	
Introduction to Object-Oriented Programming	
Course Code: CPE009B	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed:** 29/08/2024
Section: CPE21S1	**Date Submitted:** 29/08/2024
Name: Pateña, Christian Dale SJ.	**Instructor:** Engr. 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:
2.1   Identify the possible attributes and methods of a given object
2.2   Create a class using the Python language
2.3   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 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 """
2      Accounts.py
3 """
4
5 class Accounts(): # create the class
6     account_number = 0
7     account_firstname = ""
8     account_lastname = ""
9     current_balance = 0.0
10    address = ""
11    email = ""
12
13    def update_address(new_address):
14        Accounts.address = new_address
15
16    def update_email(new_email):
17        Accounts.email = new_email
```

3.Modify the Accounts.py and add *self,* before the new_address and new_email.

4.Create a new file named ATM.py and copy the code shown below:

```
1 """
2      ATM.py
3 """
4
5 class ATM():
6     serial_number = 0
7
8     def deposit(self, account, amount):
9         account.current_balance = account.current_balance + amount
10        print("Deposit Complete")
11
12    def widthdraw(self, account, amount):
13        account.current_balance = account.current_balance - amount
14        print("Widthdraw Complete")
15
16    def check_currentbalance(self, account):
17        print(account.current_balance)
```

Creating Instances of Classes

5.Create a new file named main.py and copy the code shown below:

```
1 """
2    main.py
3 """
4 import Accounts
5
6 Account1 = Accounts.Accounts() # create the instance/object
7
8 print("Account 1")
9 Account1.account_firstname = "Royce"
10 Account1.account_lastname = "Chua"
11 Account1.current_balance = 1000
12 Account1.address = "Silver Street Quezon City"
13 Account1.email = "roycechua123@gmail.com"
14
15 print(Account1.account_firstname)
16 print(Account1.account_lastname)
17 print(Account1.current_balance)
18 print(Account1.address)
19 print(Account1.email)
20
21 print()
22
23 Account2 = Accounts.Accounts()
24 Account2.account_firstname = "John"
25 Account2.account_lastname = "Doe"
26 Account2.current_balance = 2000
27 Account2.address = "Gold Street Quezon City"
28 Account2.email = "johndoe@yahoo.com"
29
30 print("Account 2")
31 print(Account2.account_firstname)
32 print(Account2.account_lastname)
33 print(Account2.current_balance)
34 print(Account2.address)
35 print(Account2.email)
```

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

```
1 """
2    main.py
3 """
4 import Accounts
5 import ATM
6
7 Account1 = Accounts.Accounts() # create the instance/object
8
9 print("Account 1")
10 Account1.account_firstname = "Royce"
11 Account1.account_lastname = "Chua"
12 Account1.current_balance = 1000
13 Account1.address = "Silver Street Quezon City"
14 Account1.email = "roycechua123@gmail.com"
15
```

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

```
31 print("Account 2")
32 print(Account2.account_firstname)
33 print(Account2.account_lastname)
34 print(Account2.current_balance)
35 print(Account2.address)
36 print(Account2.email)
37
38 # Creating and Using an ATM object
39 ATM1 = ATM.ATM()
40 ATM1.deposit(Account1,500)
41 ATM1.check_currentbalance(Account1)
42
43 ATM1.deposit(Account2,300)
44 ATM1.check_currentbalance(Account2)
45
```

9. Run the main.py program.

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 """
2     Accounts.py
3 """
4
5 class Accounts(): # create the class
6     def __init__(self, account_number, account_firstname, account_lastname,
7                  current_balance, address, email):
8         self.account_number = account_number
9         self.account_firstname = account_firstname
10        self.account_lastname = account_lastname
11        self.current_balance = current_balance
12        self.address = address
13        self.email = email
14
15    def update_address(self,new_address):
16        self.address = new_address
17
18    def update_email(self,new_email):
19        self.email = new_email
```
2. Modify the main.py and change the following codes with the red line. Do not remove the other codes in the program.

```
1  """
2      main.py
3  """
4  import Accounts
5  import ATM
6
7  Account1 = Accounts.Accounts(account_number=123456,account_firstname="Royce",
8                               account_lastname="Chua",current_balance = 1000,
9                               address = "Silver Street Quezon City",
10                              email = "roycechua123@gmail.com")
11
12 print("Account 1")
13 print(Account1.account_firstname)
14 print(Account1.account_lastname)
15 print(Account1.current_balance)
16 print(Account1.address)
17 print(Account1.email)
18
19 print()
20
21 Account2 = Accounts.Accounts(account_number=654321,account_firstname="John",
22                              account_lastname="Doe",current_balance = 2000,
23                              address = "Gold Street Quezon City",
24                              email = "johndoe@yahoo.com")
25
```

3. Run the main.py program again and run the output.

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.

Questions
1. What is a class in Object-Oriented Programming?
   A class defines the contents of the object and also identifies where the object's class or where it belongs to.
2. Why do you think classes are being implemented in certain programs while some are sequential(line-by-line)?
   The reason why we implement classes is so that presententing concepts will not be hard, avoiding redundancy on our codes that might cause confusion on our end.
3. How is it that there are variables of the same name such as account_firstname and account_lastname that exist but have different values?
   The reason why they have different values is because they have specific values or assigned to different objects to function
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?
   Constructor allows for proper initialization by providing the correct direction for the objects and their functions.

5. Explain the benefits of using Constructors over initializing the variables one by one in the main program?

The benefits of using constructors is that we can avoid the use of the same codes and redundancy would be severely reduced to the point that it will not confuse people or even ourselves while coding.

7. Conclusion:

At the end of the activity I learned how to use constructor functions properly by applying the proper ways of using classes and constructors, thus making my codes readable and clean at the same time. Constructors and classes help a lot when dealing with multi variables if we are dealing with complex concepts.

8. Assessment Rubric: