



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

Laboratory Activity No. 3.1	
Introduction to Object-Oriented Programming	
Course Code: CPE103	Program: BSCPE
Course Title: Object-Oriented Programming	Date Performed: 1/25/2025
Section: 1A	Date Submitted: 1/31/2025
Name: Ampong, J-kevin L.	Instructor: Maam 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:	



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

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 + withdraw(account: Accounts, amount: int) + update_address(new_address: string) + check_currentbalance(account: Accounts) + update_email(new_email: string) + view_transactionssummary()

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**

Please refer to this link: https://colab.research.google.com/drive/1-WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=07982604-7153-4529-9a68-ee0bcaef2d4c

2. Create a Python file inside the **OOPIntro_LastName** folder named **Accounts.py** and copy the code shown below:



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

```
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.

Please refer to this link:

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

Please refer to this link: https://colab.research.google.com/drive/1-WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=72870d98-c2ea-4ba3-b0ed-5016da1f84b9

```
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 withdraw(self, account, amount):
13        account.current_balance = account.current_balance - amount
14        print("Withdraw 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:

Please refer to this link: https://colab.research.google.com/drive/1-WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=72870d98-c2ea-4ba3-b0ed-5016da1f84b9

[WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=0a69c2d9-ffcc-4bce-93da-8f9bdce27e14](#)



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

```
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



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

red.

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

```
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
```

Please refer to this link: https://colab.research.google.com/drive/1-WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=74de22b2-a451-456f-ae7a-307e4e26b568



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

```
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

Please refer to this link: https://colab.research.google.com/drive/1-WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=39054183-8508-4fed-8eaf-85abebe163f5

[WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=39054183-8508-4fed-8eaf-85abebe163f5](https://colab.research.google.com/drive/1-WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=39054183-8508-4fed-8eaf-85abebe163f5).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.



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

Please refer to this link: https://colab.research.google.com/drive/1-WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=0e97e611-890b-471d-82bc-853003924496



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

```
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:



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

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_transactionssummary()** method. The method should display all the transaction made in the ATM object.
please refer to this link: https://colab.research.google.com/drive/1-WRFWQ5d7Ht6_BexEZoFy355U2zHE1Is?authuser=4#scrollTo=44a886f5-9b5e-4e8d-b85a-156826ae4f24

Questions

1. What is a class in Object-Oriented Programming?
A class in Object-Oriented Programming (OOP) is a blueprint for creating objects, defining their attributes (variables) and behaviors (methods). In the given code, Accounts and ATM are classes that define account-related attributes and actions like deposits and withdrawals
2.
Why do you think classes are being implemented in certain programs while some are sequential(line-by-line)?
Classes are used when organizing data and functionality into reusable components is beneficial, such as in banking applications where multiple accounts and transactions exist. Sequential programming is used for simpler tasks where object-oriented structuring isn't necessary.
3. How is it that there are variables of the same name such account_firstname and account_lastname that exist but have different values?
Variables like account_firstname and account_lastname exist with different values because each object (instance of a class) has its own independent copy of these attributes. Account1 and Account2 are separate instances of the Accounts class, storing unique values.

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 (__init__) initializes the attributes of a class when an object is created. It is called automatically when a new instance is created, ensuring that each object starts with defined values, such as an account's name, number, and balance.



UNIVERSITY OF CALOOCAN CITY
Caloocan, 1400 Metro Manila, Philippines

COLLEGE OF ENGINEERING
Computer Engineering
2nd Semester, School Year 2024-2025

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

~~Using a constructor simplifies object creation by ensuring all attributes are initialized at once, reducing redundancy and errors. Without constructors, each variable would have to be manually set after creating an object, making the code longer and harder to maintain.~~

7. Conclusion:

The implementation of classes in the provided code enhances modularity and reusability, making it easier to manage account-related actions and transactions. The constructor function plays a crucial role in initializing class attributes, ensuring each account or ATM instance has a predefined structure.

By utilizing object-oriented principles, the program maintains a clean and structured approach to handling banking transactions. This method significantly improves code maintainability compared to a sequential approach, especially when dealing with multiple objects and dynamic data.

8. Assessment Rubric: