| Laboratory Activity No. 1 | |
|---|---|
| **Introduction to Object-Oriented Programming** | |
| **Course Code:** CPE 009B | **Program:** BSCPE |
| **Course Title:** Object-Oriented Programming | **Date Performed:** 09 / 14 / 2024 |
| **Section:** CPE21S4 | **Date Submitted:** 09 / 16 / 2024 |
| **Name:** ROALLOS, Jean Gabriel Vincent G. | **Instructor:** Prof. Maria Rizette Sayo |

**5. Output of Procedure**

```
[1]  class Accounts():
       account_number = 0
       account_firstname = ""
       account_lastname = ""
       current_balance = 0.0
       address = ""
       email = ""

       def update_address(new_address):
         Accounts.address = new_address

       def update_email(new_email):
         Accounts.email = new_email
```

```
[2]  class ATM():
       serial_number = 0

       def deposit(self, account, amount):
         account.current_balance = account.current_balance + amount
         print("Deposit Complete.")

       def withdraw(self, account, amount):
         account.current_balance = account.current_balance - amount
         print("Withdraw Complete.")

       def check_currentbalance(self, account):
         print(account.current_balance)
```

```
[16] Account1 = Accounts()

     print("Account 1")
     Account1.account_firstname = "Royce"
     Account1.account_lastname = "Chua"
     Account1.current_balance = 1000
     Account1.address = "Silver Street Quezon City"
     Account1.email = "roycechua123@gmail.com"

     print(Account1.account_firstname)
     print(Account1.account_lastname)
     print(Account1.current_balance)
     print(Account1.address)
     print(Account1.email)

     print()

     Account2 = Accounts()
     Account2.account_firstname = "John"
     Account2.account_lastname = "Doe"
     Account2.current_balance = 2000
     Account2.address = "Gold Street Quezon City"
     Account2.email = "johndoe@yahoo.com"

     print("Account 2")
     print(Account2.account_firstname)
     print(Account2.account_lastname)
     print(Account2.current_balance)
     print(Account2.address)
     print(Account2.email)
```

```
Account 1
Royce
Chua
1000
Silver Street Quezon City
roycechua123@gmail.com

Account 2
John
Doe
2000
Gold Street Quezon City
johndoe@yahoo.com
```

```
[17] ATM1 = ATM()
     ATM1.deposit(Account1, 500)
     ATM1.check_currentbalance(Account1)

     ATM1.deposit(Account2, 300)
     ATM1.check_currentbalance(Account2)
```

```
Deposit Complete.
1500
Deposit Complete.
2300
```

```
[18] class Accounts():
       def __init__(self, account_number, account_firstname, account_lastname,
                    current_balance, address, email):
         self.account_number = account_number
         self.account_firstname = account_firstname
         self.account_lastname = account_lastname
         self.current_balance = current_balance
         self.address = address
         self.email = email

       def update_address(new_address):
         Accounts.address = new_address

       def update_email(new_email):
         Accounts.email = new_email
```

```
[20] Account1 = Accounts(123456, "Royce", "Chua", 1000, "Silver Street Quezon City",
                         "roycechua123@gmail.com")

     print("Account 1")
     print(Account1.account_firstname)
     print(Account1.account_lastname)
     print(Account1.current_balance)
     print(Account1.address)
     print(Account1.email)

     print()

     Account2 = Accounts(654321, "John", "Doe", 2000, "Gold Street Quezon City",
                         "johndoe@yahoo.com")

     print("Account 2")
     print(Account2.account_firstname)
     print(Account2.account_lastname)
     print(Account2.current_balance)
     print(Account2.address)
     print(Account2.email)
```

```
Account 1
Royce
Chua
1000
Silver Street Quezon City
roycechua123@gmail.com

Account 2
John
Doe
2000
Gold Street Quezon City
johndoe@yahoo.com
```

## 6. Supplementary Activity

*Modify the ATM.py program and add the constructor function.*

```
[32] class ATM():
        def __init__(self, serial_number):
          self.serial_number = serial_number

        def deposit(self, account, amount):
          account.current_balance = account.current_balance + amount
          print("Deposit Complete.")

        def withdraw(self, account, amount):
          account.current_balance = account.current_balance - amount
          print("Withdraw Complete.")

        def check_currentbalance(self, account):
          print(account.current_balance)

        def check_serialnumber(self, account):
          print(self.serial_number)
```

*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.*

```
[40] ATM1 = ATM(11111)
     ATM2 = ATM(22222)

     ATM1.deposit(Account1, 500)
     ATM1.check_currentbalance(Account1)
     ATM1.check_serialnumber()

     print()

     ATM2.deposit(Account2, 300)
     ATM2.check_currentbalance(Account2)
     ATM2.check_serialnumber()
```

```
Deposit Complete.
3000
11111

Deposit Complete.
2900
22222
```

**Questions:**

1. **What is a class in Object-Oriented Programming?**

   *A class is a container of variables and processes applicable to attributes connected to an object.*

2. **Why do you think classes are being implemented in certain programs while some are sequential(line-by-line)?**

   *Classes help group up processes and attributes that we associate to an object. By using classes, we can easily sort out and identify processes we need for certain attributes and also not repeat multiple similar processes.*

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?**

   *We can have variables with similar names, as long as we do not forget what kind of values we want it to have. Variable names should have discernable names related to its use or the value that we want to store in it.*

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 initialization of the attributes in a class helps to determine what attributes are with the object, and what values are to be expected within these attributes. It also helps in computational processes that the coder may want to do with the values within the variables.*

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

   We can prevent redundancy of code lines and we can also make pre-made processes within each class so we can just easily access it.

**7. Conclusion**

I was reviewed with the parts of a class, how it functions and it is used. This knowledge can be useful and be implemented in future Python-based projects form a much more optimized coding setup.