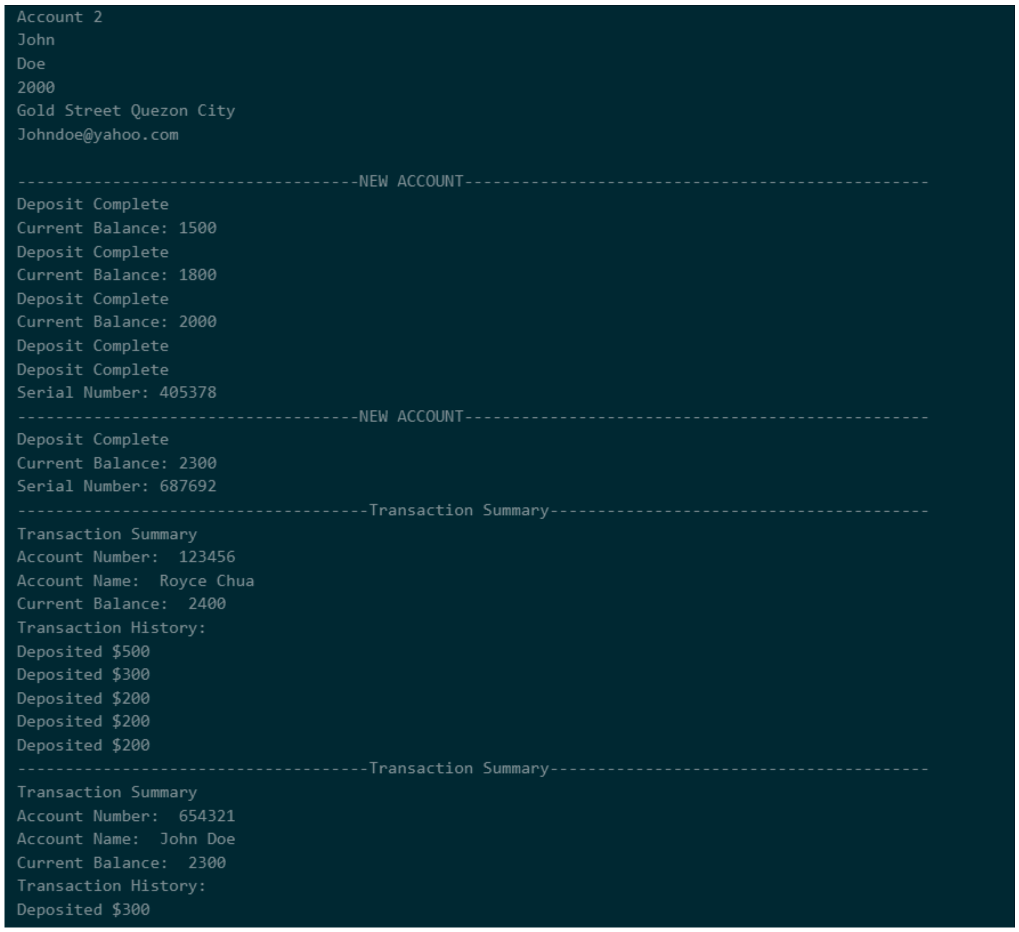
| Laboratory Activity # 1 | |
|---|---|
| **Introduction to Object-Oriented Programming** | |
| **Course Code:** CPE009B | **Program:** BSCPE |
| **Course Title:** Object-Oriented Programming | **Date Performed:** 09-14-2024 |
| **Section:** CPE21S4 | **Date Submitted:** 09-14-2024 |
| **Name(s):** Masangkay, Frederick D. | **Instructor:** Ma’am Maria Rizette Sayo |

## 6. Output



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

------------------------------------NEW ACCOUNT-------------------------------------------------
Deposit Complete
Current Balance: 1500
Deposit Complete
Current Balance: 1800
Deposit Complete
Current Balance: 2000
Deposit Complete
Deposit Complete
Serial Number: 405378
------------------------------------NEW ACCOUNT-------------------------------------------------
Deposit Complete
Current Balance: 2300
Serial Number: 687692
-------------------------------------Transaction Summary----------------------------------------
Transaction Summary
Account Number:  123456
Account Name:  Royce Chua
Current Balance:  2400
Transaction History:
Deposited $500
Deposited $300
Deposited $200
Deposited $200
Deposited $200
-------------------------------------Transaction Summary----------------------------------------
Transaction Summary
Account Number:  654321
Account Name:  John Doe
Current Balance:  2300
Transaction History:
Deposited $300
```

## 7. Supplementary Activity

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

```
"""
   ATM.py
"""



class ATM:
   def __init__ (self, serial_number = 0, account = None, amount = 0):
      self.serial_number = serial_number
      self.account = account
      self.amount = amount



   def deposit(self, amount):
      self.account.current_balance = self.account.current_balance + amount
      self.account.transaction_history.append(f"Deposited ${amount}")
      print("Deposit Complete")


   def withdraw(self, amount):
      self.account.current_balance = self.account.current_balance - amount
      self.account.transaction_history.append(f"Withdrew ${amount}")
      print("Withdraw Complete")


   def check_current_balance(self):
      print (self.account.current_balance)


   def view_transactionsummary (self):
      print("Transaction Summary")
      print("Account Number: ", self.account.account_number)
      print("Account Name: ", self.account.account_firstname, self.account.account_lastname)
      print("Current Balance: ", self.account.current_balance)
      print("Transaction History:")
      for transaction in self.account.transaction_history:
          print(transaction)
```

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.

```
"""
    Main.py
"""


import Accounts
import ATM
import random


# create the instance/object
Account1 = Accounts.Accounts(
account_number =123456,
account_firstname = "Royce",
account_lastname = "Chua",
current_balance = 1000,
address = "Silver Street Quezon City",
email = "roycechua123@gmil.com"
)


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


print()


# create the instance/object
Account2 = Accounts.Accounts(
account_number = 654321,
account_firstname = "John",
account_lastname = "Doe",
current_balance = 2000,
address = "Gold Street Quezon City",
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)


print()


# creating and Using ATM object
ATM1 = ATM.ATM(serial_number = random.randrange(10**5, 10**6), account=Account1)
ATM1.deposit(500)
print(f"Current Balance: {ATM1.check_current_balance()}")
ATM1.deposit(300)
print(f"Current Balance: {ATM1.check_current_balance()}")
ATM1.deposit(200)
print(f"Current Balance: {ATM1.check_current_balance()}")
ATM1.deposit(200)
ATM1.deposit(200)


ATM2 = ATM.ATM(serial_number = random.randrange(10**5, 10**6), account=Account2)
ATM2.deposit(300)
print(f"Current Balance: {ATM2.check_current_balance()}")


print("Serial Number:")
print(ATM1.serial_number)


ATM1.view_transactionsummary()
```

ATM2.view_transactionsummary()

3. Modify the ATM.py program and add the view_transactionsummary() method. The method should display all the transaction made in the ATM object.

```
def view_transactionsummary (self):
    print("Transaction Summary")
    print("Account Number: ", self.account.account_number)
    print("Account Name: ", self.account.account_firstname, self.account.account_lastname)
    print("Current Balance: ", self.account.current_balance)
    print("Transaction History:")
    for transaction in self.account.transaction_history:
        print(transaction)
```

**Questions:**

1. What is a class in Object-Oriented Programming?
    - A programming paradigm that uses "objects" to create computer programs and applications. It makes use of multiple methods from earlier established paradigms, such as encapsulation, modularity, and polymorphism.

2. Why do you think classes are being implemented in certain programs while some are sequential(line-by-line)?
    - Because it depends on the need and goal of the program. For complex programs, the use of classes made the code clean and easy to understand. The program can also be use many times instead of writing it over and over again.

3. How is it that there are variables of the same name such account_firstname and account_lastname that exist but have different values?
    - because we set its own attributes instead of directly assigning a value to the variable. It is simply as attributes belong to different objects therefore it has different 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 role of constructor is to have a class that can be used to construct an object. By using a constructor, setting attributes to each object was just simply assigning the value and not to the direct variable name. It can be called when adding a new object that has different values in which methods can be applied to it as well.

5. Explain the benefits of using Constructors over initializing the variables one by one in the main program?
    - It makes the code readable and reusable. It also ensures that the variable was initialized and we don't have to repeat it over and over again in another class. Constructors minimize the chance of mistakes and inconsistencies by centralizing the initialization logic and enforcing rules and constraints.

## 8. Conclusion

In this activity, I learned about the practical side of using constructors, and OOP in general. The hassle of creating a constructor and initializing its value was little compared to doing it manually or declaring it in procedural programming. I learned the hard way of the importance of attention to detail in method implementation to ensure correct functionality as I have had a lot of trouble on running the code. After all, it was a good activity to get a good grasp on handling, initializing, and passing of data into another class. It also opens my mind into the use of import, making it able to 'import' a file into the main file in which I don't have to have only one file containing everything. It's a great way to organize my code so that I can understand where my bugs are coming.

## 9. Assessment Rubric