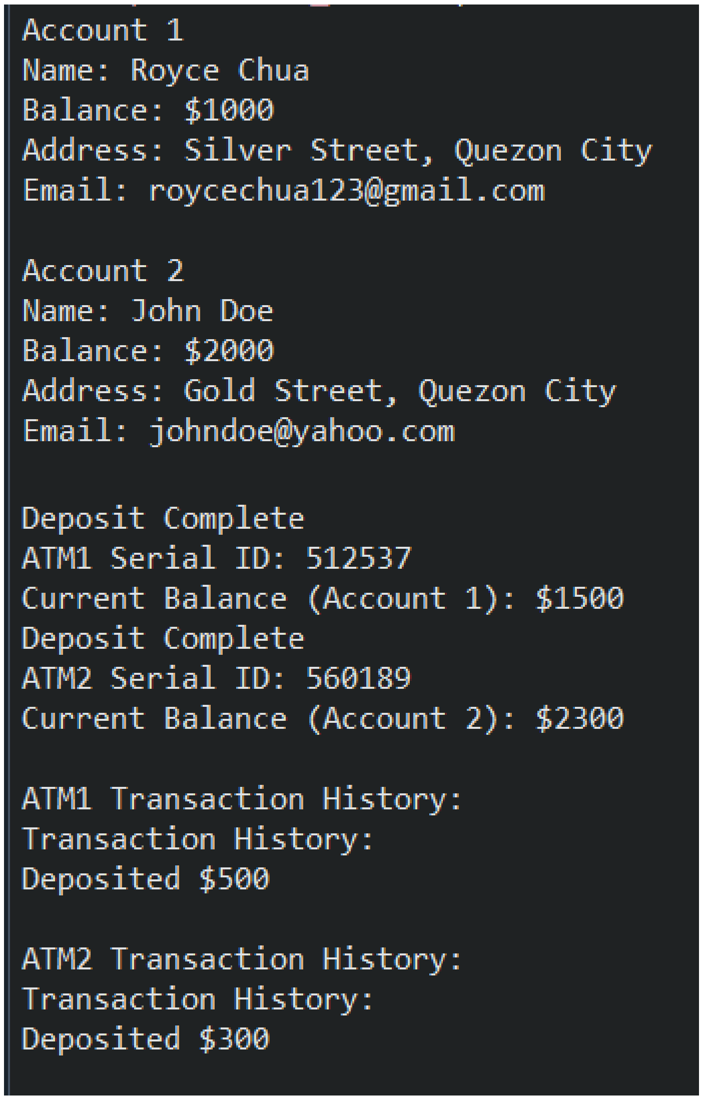
Laboratory Activity No. 1	
Introduction to Object-Oriented Programming	
Course Code: CPE009B	**Program:** BSCPE
Course Title: Object – Oriented Programming	**Date Performed:** 09/15/24
Section: CPE21s4	**Date Submitted:** 09/15/24
Name: Zolina, Anna Marie L.	**Instructor:** Mrs. Maria Rizette Sayo
6. Supplementary Activity:	

Tasks

Output of the task:



```
Account 1
Name: Royce Chua
Balance: $1000
Address: Silver Street, Quezon City
Email: roycechua123@gmail.com

Account 2
Name: John Doe
Balance: $2000
Address: Gold Street, Quezon City
Email: johndoe@yahoo.com

Deposit Complete
ATM1 Serial ID: 512537
Current Balance (Account 1): $1500
Deposit Complete
ATM2 Serial ID: 560189
Current Balance (Account 2): $2300

ATM1 Transaction History:
Transaction History:
Deposited $500

ATM2 Transaction History:
Transaction History:
Deposited $300
```

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

```
class ATM:
    def __init__(self, serial_number=0, user_account=None):
        self.serial_number = serial_number
        self.user_account = user_account
        self.transactions = []

    def deposit(self, amount):
        if amount > 0:
            self.user_account.balance += amount
            self.transactions.append(f"Deposited ${amount}")
            print("Deposit Complete")
        else:
            print("Invalid deposit amount")

    def withdraw(self, amount):
        if 0 < amount <= self.user_account.balance:
            self.user_account.balance -= amount
            self.transactions.append(f"Withdrew ${amount}")
            print("Withdrawal Complete")
        else:
            print("Insufficient funds or invalid amount.")

    def check_current_balance(self):
        return self.user_account.balance

    def view_transactions(self):
        if self.transactions:
            print("Transaction History:")
            for transaction in self.transactions:
                print(transaction)
        else:
            print("No transactions yet.")
```

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.**

```
import random
import Accounts
import ATM

account1 = Accounts.Account(account_number=123456, first_name="Royce", last_name="Chua",
                            balance=1000, address="Silver Street, Quezon City",
                            email="roycechua123@gmail.com")

print("Account 1")
print(f"Name: {account1.first_name} {account1.last_name}")
print(f"Balance: ${account1.balance}")
print(f"Address: {account1.address}")
print(f"Email: {account1.email}")

print()

account2 = Accounts.Account(account_number=654321, first_name="John", last_name="Doe",
                            balance=2000, address="Gold Street, Quezon City",
                            email="johndoe@yahoo.com")

print("Account 2")
print(f"Name: {account2.first_name} {account2.last_name}")
print(f"Balance: ${account2.balance}")
print(f"Address: {account2.address}")
print(f"Email: {account2.email}")

print()

atm1 = ATM.ATM(serial_number=random.randint(100000, 999999), user_account=account1)
atm2 = ATM.ATM(serial_number=random.randint(100000, 999999), user_account=account2)

atm1.deposit(500)
print(f"ATM1 Serial ID: {atm1.serial_number}")
print(f"Current Balance (Account 1): ${atm1.check_current_balance()}")

atm2.deposit(300)
print(f"ATM2 Serial ID: {atm2.serial_number}")
print(f"Current Balance (Account 2): ${atm2.check_current_balance()}")

print("\nATM1 Transaction History:")
atm1.view_transactions()

print("\nATM2 Transaction History:")
atm2.view_transactions()
```

**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_transactions(self):
        if self.transactions:
            print("Transaction History:")
            for transaction in self.transactions:
                print(transaction)
        else:
            print("No transactions yet.")
```

Questions:

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

   - It is a layout for creating objects. It is the one to define a set of data and functions that the objects created from the class will have. It is also the one to help the user to outline the structure they want. Class can also have subclasses that supports the main function of the given code.

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

   - It helps to structure the code and makes it easier to manage and extend. As an example, in the banking system provided in this module, you use classes to manage accounts and transactions because they contain various properties and interact with one another in complex ways. Simple scripts or jobs, on the other hand, may run line-by-line without this level of organization. It's like using a thorough blueprint for a huge structure instead of a simple set of instructions for assembling a table.

3. **How is it that there are variables of the same name such account_firstname and account_lastname that exist but have different values?**

   - Even if variables have the same name, they can appear in different situations. In OOP, every object has its own collection of variables. So, if you have two accounts, Account1 and Account2, both can contain account_firstname and account_lastname, but their values will differ. It's like having multiple people named John, but with distinct last names and personal information. They both have their unique functions, hence they have differences in certain aspects.

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 sets up the object's initial values. The constructor function is called automatically when a new instance of the class is created, ensuring correct initialization without the need for manual configuration.

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

   - Constructors make the process of generating and initializing objects more efficient, consistent, and encapsulated, resulting in fewer errors and increased readability. They simplify the code by declaring all required properties and initial values in one location, reducing repetitive code and errors. This separation of concerns enhances code readability and maintenance.

7. Conclusion

- In conclusion, I learnt how to design a banking system using Python and OOP, including how to use classes that mimic real-world things and efficiently manage code. I've learned the value of data and methods within classes by utilizing constructors to initialize objects and defining methods to control various properties. I also learnt about error handling; by arranging my code structure, I was able to track all of the defective codes that exist in the supplied tasks. This helped me realize structures that can be useful for designing programs that contain a large amount of data.

8. Assessment Rubric