Activity Name #1 - Laboratory Activity 1 (Class, Objects, Methods)	
Fernandez, Don Eleazar T.	09/14/2024
CPE 009B/CPE21S4	Maria Rizette Sayo

All written codes are below this document.

5. Procedure

OOPIntro_Fernandez > main.py > ...

```python
"""
    main.py
"""
import Accounts
import ATM

Account1 = Accounts.Accounts(account_number = 123456,
                             account_firstname = "Royce",
                             account_lastname = "Chua",
                             current_balance = 1000,
                             address = "Silver Street Quezon City",
                             email = "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.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)
```

OOPIntro_Fernandez > main.py > ...

```python
                             account_lastname = "Chua",
                             current_balance = 1000,
                             address = "Silver Street Quezon City",
                             email = "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.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)

#Creating and Using an ATM object
ATM1 = ATM.ATM()
ATM1.deposit(Account1, 500)
ATM1.check_currentbalance(Account1)

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

OOPIntro_Fernandez > Accounts.py > ...

```python
"""
    Accounts.py
"""

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(self, new_address):
        self.address = new_address

    def update_email(self, new_email):
        self.email = new_email
```

OOPIntro_Fernandez > ATM.py > ATM > check_currentbalance

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

class ATM():
    serial_number = 0

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

    def widthdraw(self, account, amount):
        account.current_balance = account.current_balance - amount
        print("Widthdraw Complete")

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

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

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

Account 2
John
Doe
2000
Gold Street Quezon City
johndoe@yahoo.com
Deposite Complete
1500
Deposite Complete
2300
```

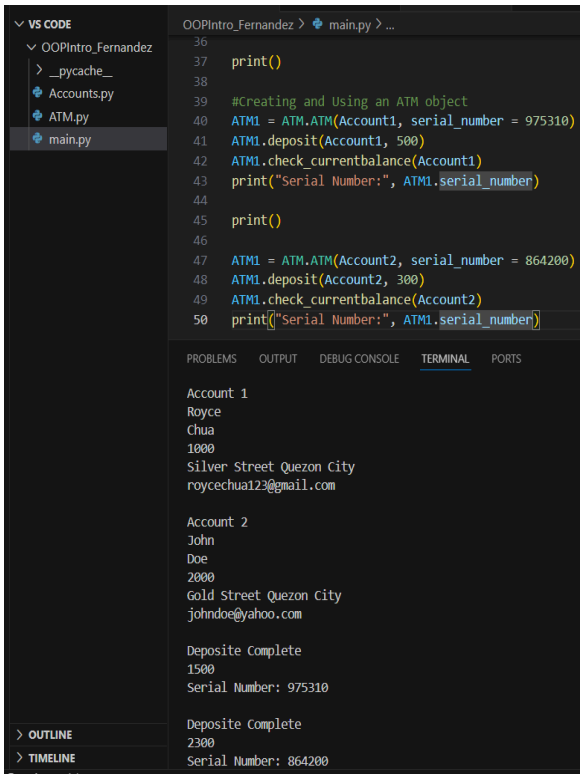
6. Supplementary Activity:

Tasks

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

```
VS CODE
OOPIntro_Ferna...
__pycache__
Accounts.py
ATM.py 8
main.py

OOPIntro_Fernandez > ATM.py > ATM > view_transactionsummary
"""
    ATM.py
"""

class ATM():
    def __init__(self, account, serial_number):
        self.account = account
        self.serial_number = serial_number

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

    def widthdraw(self, account, amount):
        account.current_balance = account.current_balance - amount
        print("Widthdraw Complete")

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

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 transactions made in the ATM object.

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

class ATM():
    def __init__(self, account, serial_number):
        self.account = account
        self.serial_number = serial_number
        self.transaction_history = []

    def deposit(self, account, amount):
        account.current_balance = account.current_balance + amount
        transaction = f"Deposited {amount} to account {account.account_number}. New Balance: {account.current_balance}"
        self.transaction_history.append(transaction)
        print("Deposite Complete")

    def widthdraw(self, account, amount):
        account.current_balance = account.current_balance - amount
        transaction = f"Withdrew {amount} from account {account.account_number}. New Balance: {account.current_balance}"
        self.transaction_history.append(transaction)
        print("Widthdraw Complete")

    def check_currentbalance(self, account):
        data = account.current_balance
        transaction = f"Checked a balance of {data} from account {account.account_number}."
        self.transaction_history.append(transaction)
        print(data)

    def view_transactionsummary(self):
        if not self.transaction_history:
            print("No Transaction.")
        else:
            print("Transaction Summary:")
            for transaction in self.transaction_history:
                print(transaction)
```

```python
class ATM():

    def view_transactionsummary(self):
        if not self.transaction_history:
            print("No Transaction.")
        else:
            print("Transaction Summary:")
            for transaction in self.transaction_history:
                print(transaction)
```

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

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

Deposite Complete
1500
Serial Number: 975310
Transaction Summary:
Deposited 500 to account 123456. New Balance: 1500
Checked a balance of 1500 from account 123456.

Deposite Complete
2300
Serial Number: 864200
Transaction Summary:
Deposited 300 to account 654321. New Balance: 2300
Checked a balance of 2300 from account 654321.
```

Questions

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

- A class defines the characteristics and actions that the objects made from it will have. In general, a class is a collection of data and functions that manage that data into one unit of code that may be reused to produce multiple objects.

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

- A class is implemented in a program for reusability, and the complexity of the whole program is to be sorted to work well with other blocks of code. Whereas sequential is simpler and works well for small tasks or one time scripts where reusability of data does not matter.

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?

- It is to sort the data of each variable to where a specific class that they belong to. This is also to point out where the data is supposed to go. For example, the words "firstname" and "lastname" have different data, but the word "account" has set them both to a class called account.

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?

- A constructor may initialize numerous data for the different functions of a particular class. It also provides flexibility to create objects with different sets of data. A constructor executes when code from the main block of code has passed a certain value to a class in which a constructor initializes.

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

- The utilization of constructors makes the code easier to read and reduces errors. They ensure objects are properly and easily initialized, which is especially useful in more complex programs.

Conclusion:

In general, this activity has helped understand the characteristics of a class. It also showed me how constructors can be utilized within it. What has made me fascinated was the use of transaction history, in which it records the activity of a user, which is placed into a constructor every time that an input activity within the atm occurs. The use of a constructor is crucial to the structure and functionality of complex codes.

Honor of pledge:

I affirm that I will not give or receive any unauthorized help on this activity/exam and that all work will be my own.

Code:

```
"""
    main.py
"""
import Accounts
import ATM

Account1 = Accounts.Accounts(account_number = 123456,
                             account_firstname = "Royce",
                             account_lastname = "Chua",
                             current_balance = 1000,
                             address = "Silver Street Quezon City",
                             email = "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.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 an ATM object
ATM1 = ATM.ATM(Account1, serial_number = 975310)
ATM1.deposit(Account1, 500)
ATM1.check_currentbalance(Account1)
print("Serial Number:", ATM1.serial_number)
ATM1.view_transactionsummary()

print()

ATM1 = ATM.ATM(Account2, serial_number = 864200)
ATM1.deposit(Account2, 300)
ATM1.check_currentbalance(Account2)
print("Serial Number:", ATM1.serial_number)
ATM1.view_transactionsummary()
```

```python
"""
    Accounts.py
"""


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(self, new_address):
        self.address = new_address

    def update_email(self, new_email):
        self.email = new_email
```

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


class ATM():
    def __init__(self, account, serial_number):
        self.account = account
        self.serial_number = serial_number
        self.transaction_history = []

    def deposit(self, account, amount):
        account.current_balance = account.current_balance + amount
        transaction = f"Deposited {amount} to account
{account.account_number}. New Balance: {account.current_balance}"
        self.transaction_history.append(transaction)
        print("Deposite Complete")

    def widthdraw(self, account, amount):
        account.current_balance = account.current_balance - amount
        transaction = f"Withdrew {amount} from account
{account.account_number}. New Balance: {account.current_balance}"
        self.transaction_history.append(transaction)
        print("Widthdraw Complete")

    def check_currentbalance(self, account):
        data = account.current_balance
        transaction = f"Checked a balance of {data} from account
{account.account_number}."
```

```
        self.transaction_history.append(transaction)
        print(data)


    def view_transactionsummary(self):
        if not self.transaction_history:
            print("No Transaction.")
        else:
            print("Transaction Summary:")
            for transaction in self.transaction_history:
                print(transaction)
```