Laboratory Activity 1 - Class, Objects, Methods	
BONIFACIO, NYKO ADREIN L.	09/15/2024
Course/Section: CpE 009B-CpE21S4	Dr. Maria Rizette Sayo

6. Supplementary Activity

TASKS

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

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

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

  def withdraw(self, account, amount):
    if account.current_balance >= amount:
      account.current_balance -= amount
      print("Withdrawal Complete")
    else:
      print("Insufficient Funds")

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

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 Accounts
import ATM

atm_machine = ATM.ATM(123456)

atm_machine.deposit(100)
atm_machine.withdraw(50)

print("ATM Information:")
print(f"Serial Number: {atm_machine.serial_number}")

```

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

```python
class ATM:
    def __init__(self, initial_balance=0):
        # Initialize any attributes here
        self.balance = initial_balance
        self.transactions = []  # List to store transactions

    def deposit(self, amount):
        self.balance += amount
        self.transactions.append(f"Deposit: ${amount}")
        print("Deposit Complete")

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient Funds")
            return
        self.balance -= amount
        self.transactions.append(f"Withdrawal: ${amount}")
        print("Withdrawal Complete")

    def view_transactionsummary(self):
        if not self.transactions:
            print("No transactions found.")
            return
        print("Transaction Summary:")
        for transaction in self.transactions:
            print(transaction)

# ... other methods and attributes here ...
```

QUESTIONS

1. What is Objected-Oriented Programming?

- In Object-Oriented Programming, a class means a structure that defines values and actions that are common to a group of related objects. It defines what (attributes) the object will possess and how these objects will be allowed to behave (operations). It is such an architectural instruction where you are directed on how to construct it and what it should resemble. For instance, Car class may have attributes such as colour and brand and behaviors like drive and brake.

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

- Programs employ classes in order to group complex data and behaviors into workable segments in order to enhance modularity, reusability and state management. They are particularly useful in case of modelling actual objects and the communications between them. Sequential programming, also referred to as line by line programming is applied where straight forward direct simplistic interpretations and simple execution are adequate for the application. The choice between the two is already made depending on the degree of task difficulty and the presence of such properties as encapsulation and scalability.

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

- The value of a variable can also depend on its scope and another variable with the same name can have a different value. Different local variables in functions or methods can have the same name as different instance attributes in different objects and even possibly global variables. For instance, account_firstname declared in the scope of a function turns out to be a different account_firstname from the one in the class or global scope. Each context therefore has a version of the variable.

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?

- In Python the constructor method __init__() is used to set values for an object's attributes at the creation of the object based on the class to which it belongs. It is used to set value to a new object Uganda for instance in ATM class values like balance or serial number are initialized. When you create an instance of the class, the constructor is invoked in order to test and set the object's properties and configurations.

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

- Using constructors ensures that all objects are consistently initialized with the required properties, lowering the likelihood of mistakes. It streamlines the code by automatically establishing variables and removing redundant initialization in the main program. Constructors also keep setup logic inside the class, resulting in better code structure and performance.

Conclusion

- Constructors play a pivotal role in the initialization of ATM objects. By automatically setting essential attributes like serial number, balance, and transaction history, they ensure consistency and prevent errors. Additionally, incorporating methods like `view_transaction_summary` into the class enhances code organization and readability, making the system more maintainable and efficient. Overall, constructors are indispensable tools for building robust and reliable ATM systems.