OOP – Lab #6

Aim: Getting Familiar with **Encapsulation** and **Public/Private** Attributes, **Instance/Class** Attributes in Python Class

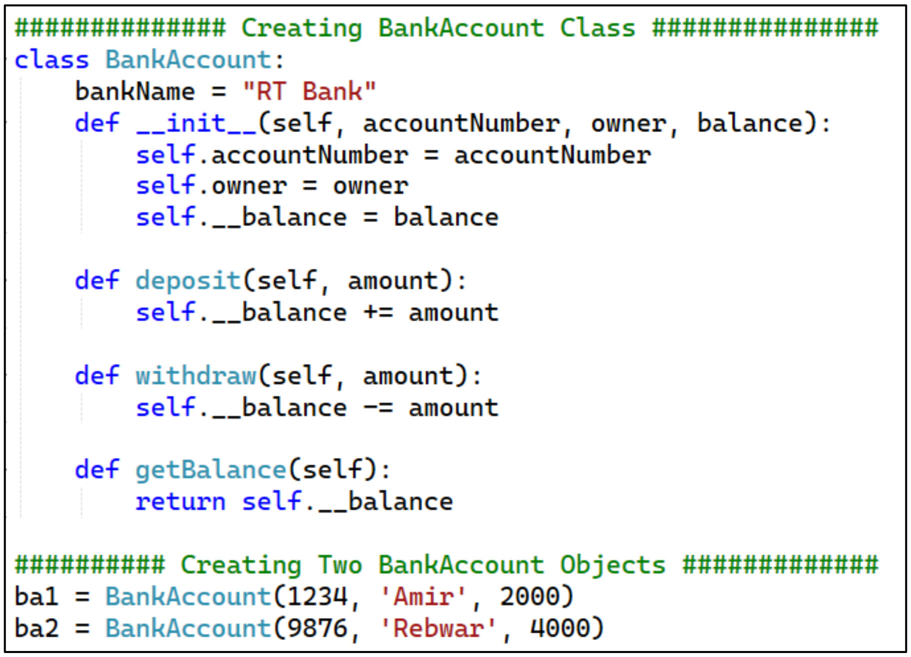
Topics:

1. **Public/Private** Attributes
2. **Instance/Class** Attributes

Lab Questions –

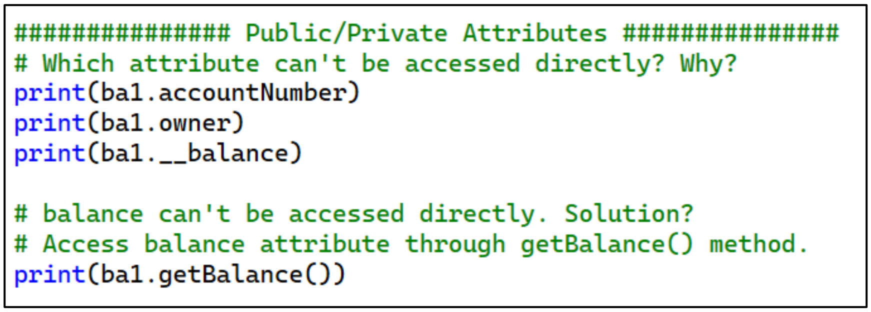
Q1 – Write Python code to create the **BankAccount** class. Then make two bank account objects.

BankAccount Class	
Class Attributes	
bankName	
Instance Attributes	
accountNumber	→ *public attribute*
owner	→ *public attribute*
balance	→ *private attribute*
Methods	
deposit(***amount***)	
withdraw(***amount***)	
getBalance( )	

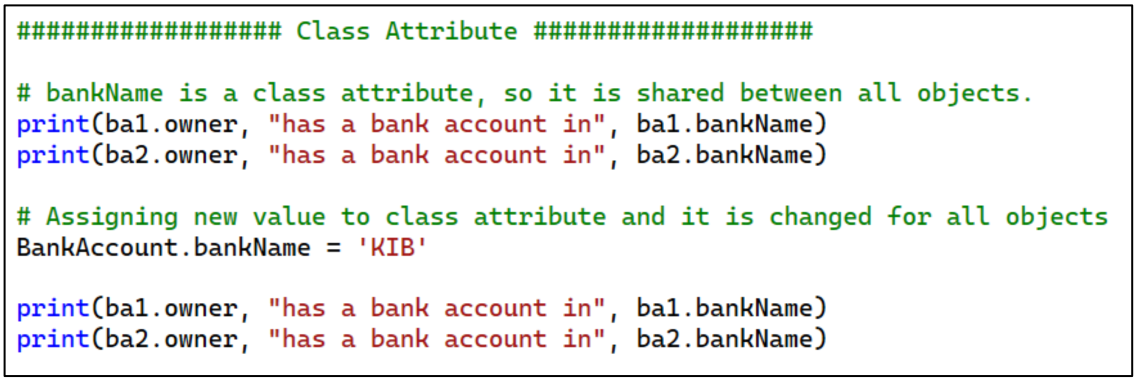


```python
############### Creating BankAccount Class ################
class BankAccount:
    bankName = "RT Bank"
    def __init__(self, accountNumber, owner, balance):
        self.accountNumber = accountNumber
        self.owner = owner
        self.__balance = balance

    def deposit(self, amount):
        self.__balance += amount

    def withdraw(self, amount):
        self.__balance -= amount

    def getBalance(self):
        return self.__balance

########## Creating Two BankAccount Objects #############
ba1 = BankAccount(1234, 'Amir', 2000)
ba2 = BankAccount(9876, 'Rebwar', 4000)
```

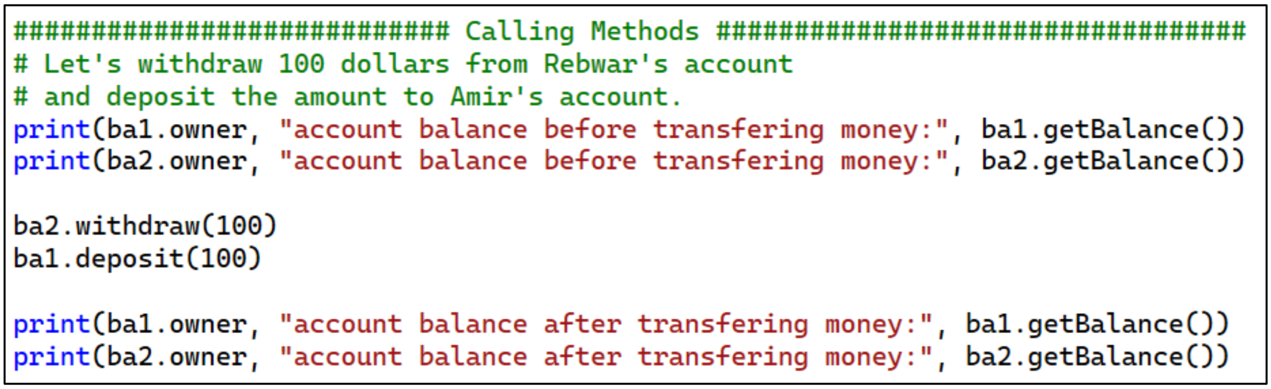
Let's try it more! Accessing private/public attributes, accessing class attribute and changing its value, and calling methods.



```
############### Public/Private Attributes ###############
# Which attribute can't be accessed directly? Why?
print(ba1.accountNumber)
print(ba1.owner)
print(ba1.__balance)

# balance can't be accessed directly. Solution?
# Access balance attribute through getBalance() method.
print(ba1.getBalance())
```

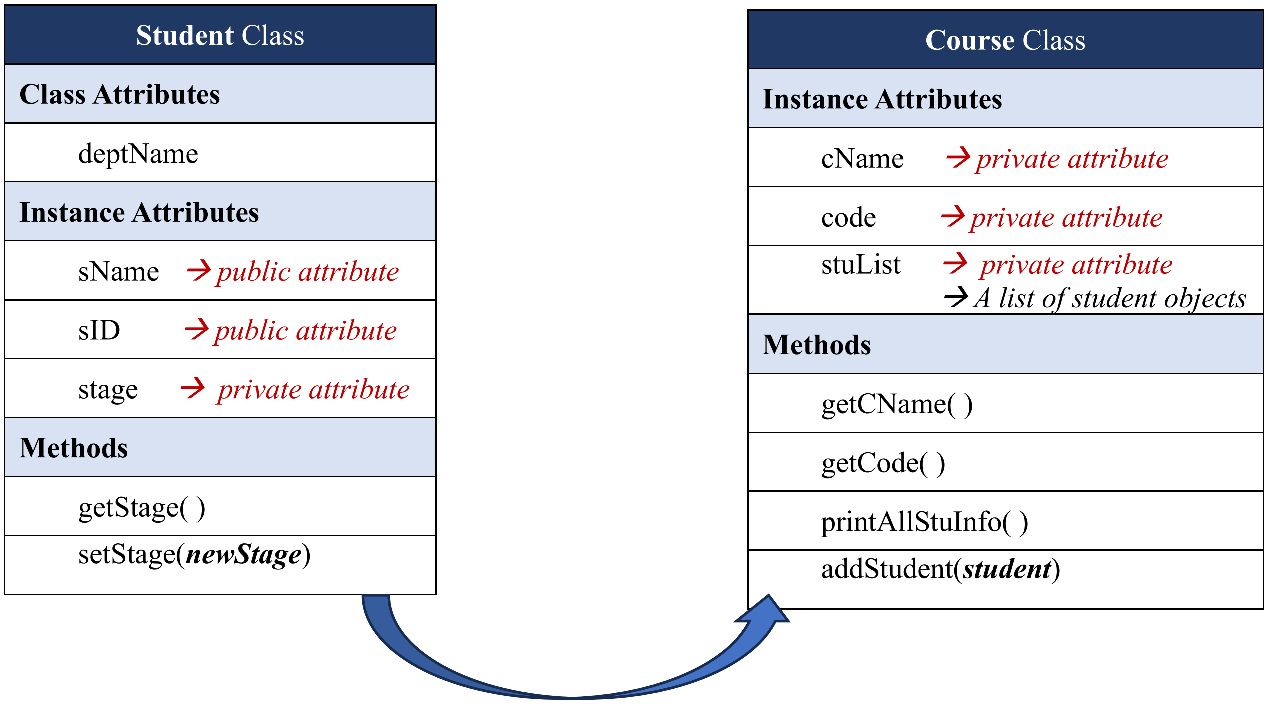


```
################## Class Attribute ###################

# bankName is a class attribute, so it is shared between all objects.
print(ba1.owner, "has a bank account in", ba1.bankName)
print(ba2.owner, "has a bank account in", ba2.bankName)

# Assigning new value to class attribute and it is changed for all objects
BankAccount.bankName = 'KIB'

print(ba1.owner, "has a bank account in", ba1.bankName)
print(ba2.owner, "has a bank account in", ba2.bankName)
```



```
############################ Calling Methods ##################################
# Let's withdraw 100 dollars from Rebwar's account
# and deposit the amount to Amir's account.
print(ba1.owner, "account balance before transfering money:", ba1.getBalance())
print(ba2.owner, "account balance before transfering money:", ba2.getBalance())

ba2.withdraw(100)
ba1.deposit(100)

print(ba1.owner, "account balance after transfering money:", ba1.getBalance())
print(ba2.owner, "account balance after transfering money:", ba2.getBalance())
```

Q2 – Write Python code to create the following two classes. Then make some objects of each class.

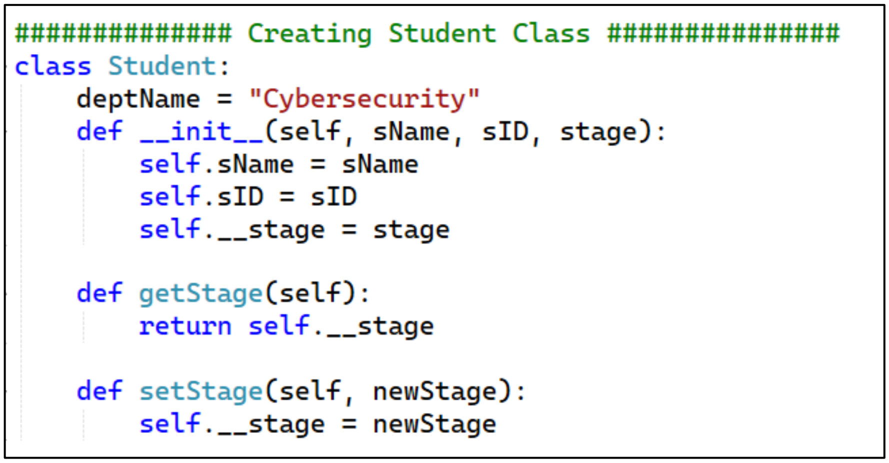


Student Class	
Class Attributes	
deptName	
Instance Attributes	
sName	→ *public attribute*
sID	→ *public attribute*
stage	→ *private attribute*
Methods	
getStage( )	
setStage(***newStage***)	

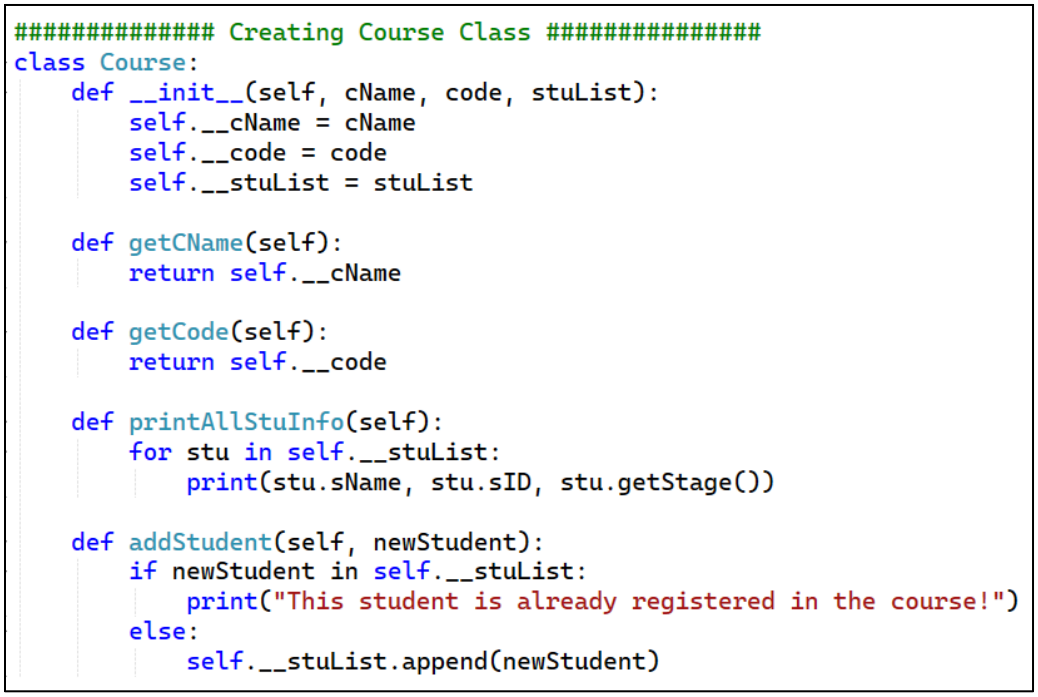
Course Class	
Instance Attributes	
cName	→ *private attribute*
code	→ *private attribute*
stuList	→ *private attribute* → *A list of student objects*
Methods	
getCName( )	
getCode( )	
printAllStuInfo( )	
addStudent(***student***)	

Many students can register for one course.

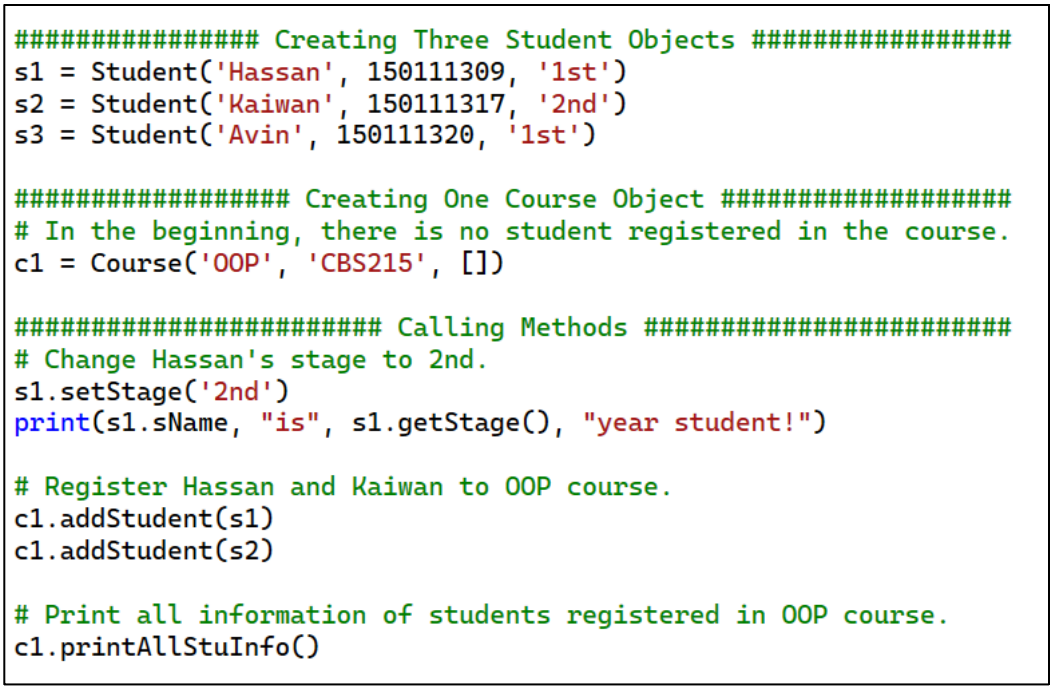
- **printAllStuInfo( )** → This method prints all information of students in the current course object.
- **addStudent(*student*)** → This method adds a student object to the **stulist**, which is a list of students registered in the current course.



```
############## Creating Student Class ###############
class Student:
    deptName = "Cybersecurity"
    def __init__(self, sName, sID, stage):
        self.sName = sName
        self.sID = sID
        self.__stage = stage

    def getStage(self):
        return self.__stage

    def setStage(self, newStage):
        self.__stage = newStage
```



```python
############## Creating Course Class ###############
class Course:
    def __init__(self, cName, code, stuList):
        self.__cName = cName
        self.__code = code
        self.__stuList = stuList

    def getCName(self):
        return self.__cName

    def getCode(self):
        return self.__code

    def printAllStuInfo(self):
        for stu in self.__stuList:
            print(stu.sName, stu.sID, stu.getStage())

    def addStudent(self, newStudent):
        if newStudent in self.__stuList:
            print("This student is already registered in the course!")
        else:
            self.__stuList.append(newStudent)
```



```python
################ Creating Three Student Objects #################
s1 = Student('Hassan', 150111309, '1st')
s2 = Student('Kaiwan', 150111317, '2nd')
s3 = Student('Avin', 150111320, '1st')

################## Creating One Course Object ###################
# In the beginning, there is no student registered in the course.
c1 = Course('OOP', 'CBS215', [])

######################## Calling Methods ########################
# Change Hassan's stage to 2nd.
s1.setStage('2nd')
print(s1.sName, "is", s1.getStage(), "year student!")

# Register Hassan and Kaiwan to OOP course.
c1.addStudent(s1)
c1.addStudent(s2)

# Print all information of students registered in OOP course.
c1.printAllStuInfo()
```