P.O.O

TD n°9 : Programmation Orientée Objet (POO)	Thème 1 : Structures de données
	COURS et EXERCICES

Compléter les classes suivantes :

- Pour la classe bancaire, suivre les consignes
- Pour la classe Calcul de moyenne, rajouter toutes les fonctions qui vous semblent pertinentes

Une class GBancaire

Un compte bancaire (simplifié) est défini par le solde disponible sur le compte. Les opérations de manipulation minimales seront :

- Initialiser un compte bancaire
- Accéder au solde d'un compte bancaire
- Créditer un compte bancaire
- Débiter un compte bancaire

On étendra également l'interface avec :

- Un constructeur permettant d'initialiser le solde.
- Une méthode `afficher` qui affiche l'objet sous le forme suivant :

```
    Titulaire : nom du titulaire
    Banque : nom de la banque
    Solde : solde du compte
```

- Une méthode `virerVers` qui effectue un virement vers un autre compte bancaire.

```
class CompteBancaire:
    def __init__(self, idNumber, nomPrenom, solde):
        self.idNumber = idNumber
        self.nomPrenom = nomPrenom
        self.solde = solde


    def versement(self, argent):
        self.solde = self.solde + argent

    def retrait(self, argent):
        if(self.solde < argent):
            print(" Impossible d'effectuer l'opération. Solde insu
        else:
            self.solde = self.solde - argent

    def agios(self):
        self.solde =self.solde*95/100

    def afficher(self):
        print("Compte numéro : " , self.idNumber)
        print("Nom & Prénom : ", self.nomPrenom)
        print(" Solde  : ", self.solde , " euros ")
        print("Sauf erreur ou omisssion ! ")

    def virervers(self,other,somme):
        self.solde-=somme
        other.solde+=somme

monCompte = CompteBancaire(16168891, " Martin Pierre", 22300)
monCompte2 = CompteBancaire(36, " Dupond Jean", 300)
monCompte.versement(1500)
monCompte.retrait(24000)
monCompte.agios()
monCompte.afficher()
monCompte.virervers(monCompte2,1500)
monCompte2.afficher()
```

```
 Impossible d'effectuer l'opération. Solde insuffisant !
Compte numéro :  16168891
Nom & Prénom :   Martin Pierre
 Solde  :  22610.0  euros
Sauf erreur ou omisssion !
Compte numéro :  36
Nom & Prénom :   Dupond Jean
 Solde  :  1800  euros
Sauf erreur ou omisssion !
```

Calcul de moyenne

Ecrire une interface et compléter la class suivante

```
class Eleve:

    def __init__(self,nom):
        self.ListeDeMatieres={}
        self.nom=nom
        self.ListeDeNotes={}


    def Nom(self):
        return self.nom

    def Matieres(self):
        return self.ListeDeMatieres

    def Notes(self):
        return self.ListeDeNotes

    def ajouterMatiere(self,matiere,coeff):
        self.ListeDeMatieres[matiere]=coeff
        self.ListeDeNotes[matiere]=[]


    def ajouterNote(self,matiere,note):
        if matiere not in self.ListeDeMatieres:
            print("La matière n'existe pas encore, on va la rajout
            coeff=input('Donnes le coefficient de cette matière :'
            self.ajouterMatiere(matiere,coeff)
            self.ajouterNote(matiere,note)
        else:
            self.ListeDeNotes[matiere].append(note)
```

```
    def CalculMoyenneMatiere(self,matiere):
        if matiere not in self.ListeDeMatieres:
            print("La matière n'existe pas encore")
        else:
            sommeNote=0
            for note in self.ListeDeNotes[matiere]:
                sommeNote+=note
            return sommeNote/len(self.ListeDeNotes[matiere])



    def CalculMoyenneGenerale(self):
        sommeNote=0
        sommeCoeff=0
        for matiere in self.ListeDeMatieres:
            sommeNote+=float(self.CalculMoyenneMatiere(matiere))*f
            sommeCoeff+=float(self.ListeDeMatieres[matiere])
        return sommeNote/sommeCoeff

    def afficher(self):
        print(f"Les informations concernant {self.nom} sont" )
        for matiere in self.ListeDeMatieres:
            print(f"La moyenne en {matiere} est {self.CalculMoyenn
        print(f"La moyenne générale est de {self.CalculMoyenneGene

    def __repr__(self):
        ch="Les informations concernant "+self.nom+" sont : \n"
        for matiere in self.ListeDeMatieres:
            ch+="- La moyenne en "+ matiere+" est " + str(self.Cal
        ch+= "La moyenne générale est de "+str(self.CalculMoyenneG
        return ch
```

```
pierre=Eleve('Pierre')

pierre.ajouterMatiere('Maths',16)
pierre.ajouterMatiere('NSI',8)
pierre.ajouterMatiere('PC',16)
print(pierre.Matieres())
print(pierre.Notes())
pierre.ajouterNote('NSI',14)
print(pierre.Notes())
pierre.ajouterNote('NSI',12)
print(pierre.Notes())
pierre.ajouterNote('NSI',10)
print(pierre.Notes())
pierre.ajouterNote('Maths',10)
print(pierre.Notes())
```

```
pierre.ajouterNote('Maths',15)
print(pierre.Notes())

pierre.ajouterNote('PC',13)
print(pierre.Notes())

pierre.ajouterNote('PC',7)
print(pierre.Notes())
pierre.ajouterNote('Anglais',7)
pierre.Notes()
print(pierre.CalculMoyenneMatiere('NSI'))
print(pierre.CalculMoyenneMatiere('Maths'))
print(pierre.CalculMoyenneMatiere('PC'))
print(pierre.CalculMoyenneGenerale())


print(pierre)
```

```
{'Maths': 16, 'NSI': 8, 'PC': 16}
{'Maths': [], 'NSI': [], 'PC': []}
{'Maths': [], 'NSI': [14], 'PC': []}
{'Maths': [], 'NSI': [14, 12], 'PC': []}
{'Maths': [], 'NSI': [14, 12, 10], 'PC': []}
{'Maths': [10], 'NSI': [14, 12, 10], 'PC': []}
{'Maths': [10, 15], 'NSI': [14, 12, 10], 'PC': []}
{'Maths': [10, 15], 'NSI': [14, 12, 10], 'PC': [13]}
{'Maths': [10, 15], 'NSI': [14, 12, 10], 'PC': [13, 7]}
La matière n'existe pas encore, on va la rajouter :


Donnes le coefficient de cette matière : 6


12.0
12.5
10.0
10.826086956521738
Les informations concernant Pierre sont :
- La moyenne en Maths est 12.5 pour un coefficient de 16
- La moyenne en NSI est 12.0 pour un coefficient de 8
- La moyenne en PC est 10.0 pour un coefficient de 16
- La moyenne en Anglais est 7.0 pour un coefficient de 6
La moyenne générale est de 10.826086956521738
```