Exercice - Programmation Orientée Objet (POO)

1- PRELIMINAIRE :

Ouvrir un nouveau fichier nommé *compteBancaire.py*. La première ligne sera la suivante :

```
from datetime import datetime
```

Exécuter le fichier afin de pouvoir disposer de la classe *datetime* de python. Exécuter ensuite dans la console la méthode *now()* disponible dans cette classe :

```
>>> print(datetime.now())
2024-10-14 21:48:09.608859

>>> datetime.now()
datetime.datetime(2024, 10, 14, 21, 48, 19, 769951)
```

On constate que cette méthode renvoie selon un format donné, l'instant précis où la commande a été exécutée. On s'en servira dans la suite de cette activité.

2- ATTRIBUTS ET CREATION DE LA METHODE CONSTRUCTEUR :

On se propose de créer une classe nommée *CompteBancaire*.

Cette classe possèdera 2 attribut de classe nommés :

- nbComptes : donne le nombre d'instances créées (*int*)
- clients : dictionnaire dont les clés sont les noms des titulaires des comptes créés et les valeurs, l'instance de classe associée qui est créée (*dictionnaire*).

Les attributs d'instances des objets créés seront :

- nom : donne le nom du titulaire du compte (*string*)
- solde : donne le solde en € du compte (*float*)
- historique : liste qui contient des listes qui répertorient les opérations bancaires menées depuis la création du compte (*list*)

⇨ Ecrire dans le fichier précédent, le début du script de la classe *CompteBancaire* pour définir les attributs de classe et ceux d'instance à travers l'écriture de la méthode constructeur : __init__() :

Par exemple, en créant les 3 instances suivantes dans le programme principal :

```
dupont = CompteBancaire("Dupont",2000)
martin = CompteBancaire("Martin",500)
mathieu = CompteBancaire(solde = 45000 , nom = "Mathieu")
```

On peut obtenir dans la console :

```
>>> CompteBancaire.nbComptes
3
```

```
>>> CompteBancaire.clients
{'Dupont': <__main__.CompteBancaire object at 0x0000022530193910>,
'Martin': <__main__.CompteBancaire object at 0x0000022530193820>, '
Mathieu': <__main__.CompteBancaire object at 0x0000022530193A90>}
```

```
>>> dupont.nom
'Dupont'
```

```
>>> dupont.solde
2000
```

```
>>> dupont.historique
[[datetime.datetime(2024, 10, 14, 21, 40, 57, 296969), 'création compte', 2000, 2000]]
```

Instant de la création du compte

Opération réalisée

Valeur de l'opération

Solde du compte une fois l'opération réalisée

3- CREATION DE LA METHODE : __STR__() :

⇨ Compléter ce script de la classe *Comptebancaire* en écrivant celui de la méthode *__str__()* .

Une fois cette méthode écrite, on pourra vérifier son bon fonctionnement, en exécutant les lignes suivantes dans la console :

```
>>> print(dupont)
Compte bancaire de Dupont : solde de 2000 €

>>> print(mathieu)
Compte bancaire de Mathieu : solde de 45000 €
```

4- CREATION DE LA METHODE : VERSEMENT() :

⇨ Compléter le script de la méthode *versement()* qui prend en argument une somme d'argent qui sera crédité sur le compte.

Une fois cette méthode écrite, on pourra vérifier son bon fonctionnement, en exécutant les lignes suivantes dans la console :

```
>>> mathieu.versement(100000)

>>> print(mathieu)
Compte bancaire de Mathieu : solde de 145000 €

>>> mathieu.historique
[[datetime.datetime(2024, 10, 14, 22, 3, 23, 238838), 'création compte', 45000, 45000],
[datetime.datetime(2024, 10, 14, 22, 3, 25, 735344), 'versement', 100000, 145000]]
```

5- CREATION DE LA METHODE : RETRAIT() :

⇨ Compléter le script de la méthode *retrait()* qui prend en argument une somme d'argent qui sera retirée du compte.

Une fois cette méthode écrite, on pourra vérifier son bon fonctionnement, en exécutant les lignes suivantes dans la console :

```
>>> mathieu.retrait(200000)

>>> print(mathieu)
Compte bancaire de Mathieu : solde de -55000 €

>>> mathieu.historique
[[datetime.datetime(2024, 10, 14, 22, 3, 23, 238838), 'création compte', 45000, 45000]
, [datetime.datetime(2024, 10, 14, 22, 3, 25, 735344), 'versement', 100000, 145000], [
datetime.datetime(2024, 10, 14, 22, 5, 57, 129917), 'retrait', 200000, -55000]]
```

6- CREATION DE LA METHODE : ECRITUREH() :

⇨ Compléter le script de la méthode *ecritureH()* qui ne prend aucune variable en argument et qui permet d'afficher dans la console, les informations contenues dans l'attribut *historique* de l'objet.

Une fois cette méthode écrite, on pourra vérifier son bon fonctionnement, en exécutant la ligne suivante dans la console :

```
>>> mathieu.ecritureH()
2024-10-14 22:11:34.712307 : création compte , 45000 € , solde : 45000 €
2024-10-14 22:11:43.483662 : versement , 100000 € , solde : 145000 €
2024-10-14 22:11:52.269689 : retrait , 200000 € , solde : -55000 €
```

7- CREATION DE LA METHODE : TRANSFERT() :

⇨ Compléter le script de la méthode *transfert()* qui prend en argument une autre instance de cette classe et une somme d'argent. Cette somme sera transférée du compte de l'instance de classe vers le compte de l'autre instance de classe mise en argument.

Une fois cette méthode écrite, on pourra vérifier son bon fonctionnement, en exécutant les lignes suivantes dans la console :

```
>>> mathieu.transfert(dupont,1000)

>>> mathieu.ecritureH()
2024-10-14 22:11:34.712307 : création compte , 45000 € , solde : 45000 €
2024-10-14 22:11:43.483662 : versement , 100000 € , solde : 145000 €
2024-10-14 22:11:52.269689 : retrait , 200000 € , solde : -55000 €
2024-10-14 22:16:32.250922 : virement vers compte Dupont , -1000 € , solde : -56000 €

>>> dupont.ecritureH()
2024-10-14 22:11:34.712307 : création compte , 2000 € , solde : 2000 €
2024-10-14 22:16:32.250922 : virement du compte Mathieu , 1000 € , solde : 3000 €
```

⇨ Uploader le fichier sur nsibranly.fr