

**DEVOIR DE CONTROLE DU SEMESTRE 1****Matière : INFORMATIQUE****Classes : 2^{ème} Année MP-PC-PT****Durée : 1 h**

L'indentation dans les instructions Python, la clarté de la copie seront prises en considération dans le barème.

Problème

L'objectif du problème est de créer une application de gestion de comptes bancaires des différents clients d'une banque.

Afin de réaliser cette application, nous devons définir une classe **CompteBancaire**, une classe **Client** et une classe **Banque**.

A. La classe CompteBancaire

Un compte Bancaire possède les attributs suivants : *numero_compte*, *titulaire (nom du propriétaire du compte)* et *solde*.

Définir la classe **CompteBancaire** qui contient les méthodes suivantes :

1. **__init__** qui permet d'initialiser les attributs d'objet avec des valeurs données en argument lors de l'appel de la méthode.

NB. Le paramètre solde du constructeur vaut par défaut 0.

2. **__repr__** qui retourne les différentes informations d'un compte bancaire : "Le solde du compte *numero_compte* du client *titulaire* est de *solde* Dt"

Exemples:

```
>>> comptel = CompteBancaire("TN123456", "Ala")
```

```
>>> comptel
```

```
'Le solde du compte TN123456 du client Ala est de 0 Dt'
```

```
>>> compte2 = CompteBancaire("TN654321", "Ala", 1000)
```

```
>>> compte2
```

```
'Le solde du compte TN654321 du client Ala est de 1000 Dt'
```

3. **__eq__** qui prend en paramètre *other* de type **CompteBancaire** et retourne True si toutes les valeurs des attributs des deux instances sont égaux, sinon retourne False.

Exemple:

```
>>> comptel==compte2
```

```
False
```



4. **deposer** qui prend en paramètre **m** représentant le montant à déposer dans le compte. Si **m** est positif, cette méthode ajoute **m** au solde actuel et affiche "montant Dt déposés dans le compte numero_compte. Nouveau solde : solde Dt". Dans le cas contraire cette méthode affiche "Le montant doit être positif"

Exemple:

```
>>> comptel.deposer(500)
500 Dt déposés dans le compte TN123456. Nouveau solde : 500 Dt
```

5. **retirer** qui permet de retirer un montant **m** du solde actuel et lever une exception selon les cas. Dans le cas où le montant est plus grand du solde, afficher le message "Solde insuffisant, Solde actuel: solde Dt" Si le montant est négatif, afficher "Le montant doit être supérieur à zéro. " Dans le cas contraire afficher "montant Dt retirés du compte numero_compte, Nouveau solde : solde Dt"

Exemples:

```
>>> comptel.retirer(1000)
Solde insuffisant, Solde actuel: 500Dt
>>> comptel.retirer(200)
200 Dt retirés du compte TN123456, Nouveau solde : 300 Dt
```

6. **afficher_solde** qui permet d'afficher le solde actuel du compte selon l'exemple ci-dessous.

Exemple:

```
>>> comptel.afficher_solde()
'Le solde du compte TN123456 est de 300 Dt.'
```

B. La classe Client

La classe **Client** gère les informations personnelles d'un client ainsi que les comptes bancaires qui lui sont associés. Cette classe possède les attributs **nom**, **ID_client** et **comptes** (liste des comptes bancaires, objets de la classe CompteBancaire, détenus par un client)

Définir la classe **Client** qui contient les méthodes suivantes :

7. **__init__** qui prend en paramètre le **nom** et **ID_client** du client puis initialise **comptes** en une liste vide.

Exemple:

```
>>> client1=Client("Ala",12345)
```

8. **ajouter_compte** qui prend en paramètre **compte** de type **CompteBancaire**. Si le compte existe déjà la méthode affiche le message "compte numero_compte existe déjà ". Dans le cas contraire le compte est ajouté à la liste **comptes**.

Exemples:

```
>>> client1.ajouter_compte(comptel)
>>> client1.ajouter_compte(comptel)
'compte TN123456 existe déjà'
>>> client1.ajouter_compte(compte2)
```

9. **afficher_solde_comptes** qui permet d'afficher les soldes de tous les comptes d'un client.

Indication : utiliser la méthode **afficher_solde** de la classe **CompteBancaire**

Exemples:

```
>>> client1.afficher_solde_comptes()
```



'Comptes de Ala :

Le solde du compte TN123456 est de 300 Dt.

Le solde du compte TN654321 est de 1000 Dt.'

10. déposer_dans_compte qui prend en paramètre *numero_compte* et le montant *m*. Cette méthode permet de déposer un montant *m* dans le compte de numéro égal à *numero_compte*. Si le compte est non trouvé, cette méthode affiche le message "Compte non trouvé".

Indication. On doit utiliser la méthode **deposer** de la classe **CompteBancaire**.

Exemples:

```
>>> client1.deposer_dans_compte("TN123456", 350)
'350 Dt déposés dans le compte TN123456. Nouveau solde : 650 Dt'
>>> client1.deposer_dans_compte("FR12", 350)
'Compte non trouvé'
```

11. retirer_du_compte qui prend en paramètre *numero_compte* et le montant *m*. Cette méthode permet de retirer un montant *m* du compte ayant le numéro égal à *numero_compte*. Si le compte est non trouvé, cette méthode affiche le message "Compte non trouvé".

Indication. On doit utiliser la méthode **retirer** de la classe **CompteBancaire**.

Exemples:

```
>>> client1.retirer_de_compte("TN123456", 50)
50 Dt retirés du compte TN123456, Nouveau solde : 600 Dt
>>> client1.retirer_de_compte("TN000000", 400)
Compte non trouvé
```

C. La classe Banque

La classe **Banque** permet de gérer plusieurs clients et leurs créer de nouveaux comptes. Cette classe possède un attribut *clients* de type dict dont les clés sont les identifiants des clients et les valeurs sont des objets de la classe **Client**.

Définir la classe **Banque** qui contient les méthodes suivantes :

12. __init__ qui permet d'initialiser l'attribut *clients* à un dictionnaire vide.

Exemple:

```
>>> banque = Banque()
```

13. ajouter_client qui prend en paramètre *client* de type **Client** et qui permet d'ajouter un nouveau client à la banque. Cette méthode vérifie d'abord l'existence de l'*ID_client* du *client* passé en paramètre. Si l'*ID* existe dans le dictionnaire, la méthode retourne le message suivant "Client nom existe déjà dans la banque". S'il s'agit d'un nouveau client, ce dernier est ajouté avec son *ID_client* comme clé et *client* comme valeur et la méthode retourne le message suivant "Client nom est ajouté à la banque"

Exemples:

```
>>> client2=Client("Amine", 356478)
>>> print(banque.ajouter_client(client2))
'Client Amine est ajouté à la banque'
```



14. creer_compte qui prend en paramètre *client* de type **Client**, *numero_compte* et *solde_initial* ayant une valeur par défaut égale à 0. Cette méthode permet de créer un compte pour le *client* passé en paramètre. Elle vérifie d'abord l'existence de l'identifiant du *client* dans le dictionnaire **clients**. Si *client* n'est pas un client de la banque, cette méthode retourne un message "Client nom_client non trouvé dans la banque", sinon un nouveau compte instance de la classe **CompteBancaire** lui est créé en retournant le message "Compte numero_compte créé pour nom avec un solde initial solde_initial Dt"

Indication. On doit utiliser la méthode **ajouter_compte** de la classe **Client**.

Exemple:

```
>>> print(banque.creer_compte(client2, "TN111111", 400))
'Compte TN111111 créé pour Amine avec un solde initial 400 Dt'
```

15. afficher_clients qui permet d'afficher les informations de chaque client ainsi que les comptes qui lui sont associés. Si la banque n'a pas de clients, cette méthode affiche le message "Aucun client dans la banque."

Indication. On doit utiliser la méthode **afficher_solde_comptes** de la classe **Client**.

Exemple:

```
>>> banque.afficher_clients()
'Liste des clients et leurs comptes :
Client : Amine
Comptes de Amine :
Le solde du compte TN111111 est 400 Dt'
```

16. operation qui prend en paramètre *client*, *numero_compte*, *montant*, et *type_op*. Cette méthode permet d'effectuer un dépôt dans le compte du *client* dont le numéro de compte est égal *numero_compte* si la valeur du paramètre *type_op* est égale à "depot" ou un retrait si cette dernière est égale à "retrait".

NB. Avant de faire l'operation de dépôt ou de retrait dans le compte, il faut vérifier d'abord l'existence du client dans la banque. Si *client* n'est pas un client de la banque, cette méthode affiche un message "Client nom_client non trouvé dans la banque"

Indication. On doit utiliser les méthodes **deposer_dans_compte** et **retirer_de_compte** de la classe **Client**.

Exemple:

```
>>> banque.operation(client2, "TN111111", 100, 'depot')
'100 Dt déposés dans le compte TN111111, Nouveau solde : 500 Dt'
```