OFPPT	مكتب التكوين المهني وإنعاش الشغل
	Office de la Formation Professionnelle et de la Promotion du Travail

Filière: Dev Digital 101/102

Module: Programmation Orientée Objet ***TP 10***

Exercice1 :

- Un compte bancaire possède à tout moment une donnée : solde.
- Chaque **CompteBancaire** est caractérisé par un code incrémenté automatiquement.
- A sa création, un compte bancaire a un solde nul et un code incrémenté.
- Il est aussi possible de créer un compte en précisant son solde initial.
- Utiliser son compte consiste à pouvoir y faire des dépôts, des retraits et des transferts. Pour ces trois opérations, il faut connaître le montant de l'opération.
- L'utilisateur peut aussi consulter le code et le solde de son compte par la méthode **__str__ ().**
- Un **CompteEpargne** est un compte bancaire qui possède en plus un champ « TauxInterêt» dont la valeur par défaut est 6, et une méthode **CalculIntérêt ()** qui permet de mettre à jour le solde en tenant compte des interêts (solde=solde*(1+taux/100))
- Un **ComptePayant** est un compte bancaire pour lequel chaque opération de retrait, de versement et de transfert est payante et vaut 5dh (le transfert est payant pour un compte payant)

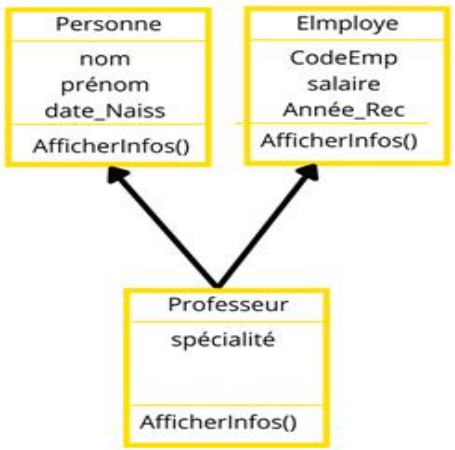
Questions :

- Définir la classe **CompteBancaire**
- Définir la classe **CompteEpargne.**
- Définir la classe **ComptePayant.**
- Dans un fichier main.py tester les classes **CompteBancaire** et **CompteEpargne** avec les actions suivantes :
  - Créer une deux instances de la classe **CompteBancaire,** deux instances de la classe **CompteEpargne** et deux instances de la classe **ComptePayant**
  - Faire appel à la méthode **verser()** de chaque instance pour déposer une somme quelconque dans ces comptes.

- Faire appel à la méthode **retirer()** de chaque instance pour retirer une somme quelconque de ces comptes.
- Faire appel à la méthode **transférer()** de chaque instance pour transférer une somme quelconque de a un autre compte.
- Faire appel à la méthode **calcul Intérêt()** du **CompteEpargne**.
- Afficher le solde des comptes précédents.

Exercice 2:

Soit le modèle suivant:



1) Créer la classe **Personne** caractérisé par le nom, le prénom et une date de naissance
2) Créer un constructeur d'initialisation permettant d'initialiser les attributs à des valeurs passé en paramètre. Les attributs sont **protected**.
3) Ajouter les accesseurs nécessaires
4) Créer la méthode **AfficherInfos**() permettant d'afficher les informations de la personne
5) Créer la méthode **__str__()** pour retourner les informations de la personne
6) Créer la classe **Employé**, caractérisé par un code auto incrémenté, un salaire et une année de recrutement.
7) Créer un constructeur d'initialisation permettant d'initialiser tous les attributs de la classe **Employe**. Les attributs sont **protected**.
8) Ajouter les accesseurs nécessaires.
9) Créer la méthode **AfficherInfos**() permettant d'afficher les informations de l'employé
10) Créer la méthode **__str__()** pour retourner les informations de **Employe**
11) Créer la classe **Professeur** qui hérite de la classe **Employe** et la classe Personne. Un professeur se caractérise en plus des attributs de la classe Personne et la classe **Employe**, par la spécialité.

12) Créer le constructeur de la classe professeur en appelant les constructeurs des classes parentes. L'attribut spécialité est privé.
13) Redéfinir la méthode AfficherInfos() pour afficher toutes les informations de la classe professeur
14) Redéfinir la méthode __str__()
15) Créer une instance de la classe Personne, et faire appel à la méthode AfficherInfos()
16) Créer une instance de la classe Employe, et faire appel à la méthode AfficherInfos()
17) Créer une instance de la classe Professeur, et faire appel à la méthode AfficherInfos()
18) Afficher les instances de chaque classe pour vérifier la méthode __str__()