TP N°3 : L'héritage

Objectif :

- **Créer une classe dérivée.**
- **Ajouter des méthodes à une classe dérivée.**
- **Redéfinir des méthodes dans une classe dérivée.**

Exercice 1 :

Classe de Base Personne :

1. Créez une classe Personne avec les attributs suivants :
   - nom : le nom de la personne.
   - age : l'âge de la personne.
2. Ajouter des méthodes d'accès pour chaque attribut
3. Ajoutez une méthode afficher_infos() qui affiche le nom et l'âge de la personne.

Classe Dérivée Etudiant :

1. Créez une classe Etudiant dérivée de la classe Personne.
2. Ajoutez un attribut supplémentaire :
   - Filière : la filière de l'étudiant.
3. Ajouter des méthodes d'accès pour l'attribut filière.
4. Redéfinissez la méthode afficher_infos() pour afficher également la matière et indiquer que la personne a le statut d'étudiant.

Classe Dérivée Employe :

1. Créez une classe Employe dérivée de la classe Personne.
2. Ajoutez un attribut supplémentaire :
   - poste : le poste occupé par l'employé.
3. Ajouter des méthodes d'accès pour l'attribut poste.
4. Redéfinissez la méthode afficher_infos() pour afficher également le poste et indiquer que la personne a le statut d'employé.

Utilisation des Classes :

Créez des instances de la classe Personne, de la classe Etudiant et de la classe Employe. Utilisez la méthode afficher_infos() pour afficher les informations spécifiques à chaque instance.

Exercice 2 :

Classe de Base Employe :

1. Créez une classe Employe avec les attributs suivants :
    - nom : le nom de l'employé.
    - salaire : le salaire de l'employé.

2. Ajouter des méthodes d'accès pour chaque attribut
3. Ajoutez une méthode calculer_salaire_annuel() qui renvoie le salaire annuel de l'employé.
4. Ajoutez une méthode __str__ qui retourne le nom et le salaire de l'employé.

Classe Directeur (Hérite de Employe) :

1. Créez une classe Directeur qui hérite de la classe Employe.
2. Ajoutez des attributs supplémentaires :
    - prime : la prime attribuée au directeur.
    - actions : le nombre d'actions que le directeur possède dans l'entreprise.

3. Ajouter des méthodes d'accès pour chaque attribut
4. Redéfinissez la méthode calculer_salaire_annuel() pour inclure la prime.
5. Redéfinissez la méthode __str__ pour inclure les informations spécifiques aux directeurs.

Utilisation des Classes :

1. Créez des instances de la classe Employe et de la classe Directeur.
2. Appelez les méthodes calculer_salaire_annuel() et affichez les informations avec la méthode __str__.

Exercice 3 :

Classe de Base Livre :

1. Créez une classe Livre avec les attributs suivants :
    - titre : le titre du livre.
    - auteur : l'auteur du livre.
    - annee_publication : l'année de publication du livre.
    - disponible : un indicateur indiquant si le livre est disponible à l'emprunt.

2. Ajouter des méthodes d'accès pour chaque attribut
3. Ajoutez une méthode emprunter() qui permet d'emprunter le livre (si disponible) et met à jour l'indicateur de disponibilité.
4. Ajoutez une méthode rendre() qui permet de rendre le livre et met à jour l'indicateur de disponibilité.
5. Ajoutez une méthode __str__ qui retourne les informations du livre.

Classe LivreNumerique (Hérite de Livre) :

1. Créez une classe LivreNumerique qui hérite de la classe Livre.
2. Ajoutez un attribut supplémentaire :
    - format : le format du livre numérique (PDF, EPUB, etc.).
3. Ajoutez une méthode telecharger() qui simule le téléchargement du livre numérique.
4. Ajoutez une méthode convertir_format(nouveau_format) qui permet de convertir le livre numérique dans un nouveau format.
5. Redéfinissez la méthode __str__ pour inclure les informations spécifiques aux livres numériques.

Utilisation des Classes :

3. Créez des instances de la classe Livre et de la classe Livre numérique.
4. Appelez les méthodes et affichez les informations avec la méthode __str__.

Exercice 4 :

Classe de Base Produit :

1. Créez une classe Produit avec les attributs suivants :
    - nom : le nom du produit.
    - prix : le prix du produit.

2. Ajouter des méthodes d'accès pour chaque attribut
3. Ajoutez une méthode __str__ qui affiche le nom et le prix du produit.
4. Ajoutez une méthode calculer_prix_final() qui renvoie le prix final du produit.

Classe ProduitElectronique (Hérite de Produit) :

1. Créez une classe ProduitElectronique qui hérite de la classe Produit.
2. Ajoutez des attributs supplémentaires :
    - marque : la marque du produit électronique.
    - garantie : la durée de garantie du produit électronique.

3. Ajouter des méthodes d'accès pour chaque attribut.
4. Ajoutez une méthode prolonger_garantie(duree) qui permet de prolonger la garantie du produit électronique.
5. Redéfinissez la méthode __str__ pour inclure les informations spécifiques aux produits éléctroniques.

Classe ProduitEnPromotion (Hérite de Produit) :

1. Créez une classe ProduitEnPromotion qui hérite de la classe Produit.
2. Ajoutez des attributs supplémentaires :
    - pourcentage_reduction : le pourcentage de réduction appliqué au produit en

promotion.

3. Ajouter des méthodes d'accès pour chaque attribut.
4. Ajoutez une méthode calculer_reduction() qui retourne la réduction du produit en promotion.
5. Redéfinissez la méthode calculer_prix_final() pour inclure la réduction.
6. Redéfinissez la méthode __str__ pour inclure les informations spécifiques aux produits en promotion.

Utilisation des Classes :

Créez des instances de la classe Produit, de la classe ProduitElectronique, et de la classe ProduitEnPromotion.

Appelez les méthodes spécifiques et affichez les informations.

Exercice 5 :

- Un compte bancaire possède à tout moment une donnée : son solde. Ce solde peut être positif (compte créditeur) ou négatif (compte débiteur).
- Chaque compte 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 et des retraits. Pour ces deux opérations, il faut connaître le montant de l'opération.
- L'utilisateur peut aussi consulter le solde de son compte par la méthode *__str__*.
- Un compte Epargne est un compte bancaire qui possède en plus un champ « TauxInterêt » et une méthode calculIntérêt() qui permet de mettre à jour le solde en tenant compte des intérêts.
- Un Compte Payant est un compte bancaire pour lequel chaque opération de retrait et de versement est payante et vaut 1 DH.

Créer un programme permettant de tester les classes CompteBancaire, CompteEpargne et ComptePayant.